Programa de Pós-Graduação em Ciência da Computação - PPGCC
Instituto de Ciência e Tecnologia - ICT
Universidade Federal de São Paulo - UNIFESP

Dissertação de Mestrado

MPQUIC-SBD: UMA IMPLEMENTAÇÃO DO PADRÃO RFC8382 PARA DETECÇÃO DE COMPARTILHAMENTO DE GARGALOS NO PROTOCOLO MPQUIC

Aluno
THOMAS WILLIAM DO PRADO PAIVA

Orientador
PROF. DR. BRUNO YUJI LINO KIMURA

Thomas William do Prado Paiva

Título da Dissertação: MPQUIC-SBD: Uma implementação do Padrão RFC8382 para Detecção de Compartilhamento de Gargalos no Protocolo MPQUIC

Dissertação apresentada à Universidade Federal São Paulo como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Área de Concentração: Sistemas da Computação / Redes de Computadores

Aprovada em 30 de abril de 2021.

Presidente da Banca:

Prof.(a) Dr.(a) Bruno Yuji Lino Kimura

Banca Examinadora:

Prof(a). Dr(a). Michele Nogueira Lima

Prof(a). Dr(a). Simone Ferlin

Prof(a). Dr(a). Roberto Sadao Yokoyama

Thomas William do Prado Paiva: *MPQUIC-SBD: Uma implementação do Padrão RFC8382 para Detecção de Compartilhamento de Gargalos no Protocolo MPQUIC*, Dissertação de Mestrado apresentada ao Instituto de Ciência e Tecnologia - UNIFESP, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, Orientador: Prof. Dr. Bruno Kimura, Abril 2021.

Paiva, Thomas
MPQUIC-SBD: Uma implementação do Padrão RFC8382 para Detecção de Compartilhamento de Gargalos no Protocolo MPQUIC / Thomas Paiva
Orientador(a) Bruno Kimura. - São José dos Campos, 2021.
80 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Mestrado em Ciência da Computação) - Universidade Federal de São Paulo - Instituto de Ciência e Tecnologia, 2021.

1. Computer Network; Transport Protocols; TCP; UDP. 2. Quick UDP Internet Connection (QUIC); Multipath QUIC (MPQUIC). 3. Shared Bottleneck Detection (SBD); RFC8382. 4. DASH (Dynamic Adaptive Streaming over HTTP); Stream Vídeo. 5. Network Measurement; Passive Measurement; Active Measurement. I. Kimura, Bruno, coorientador(a). II. Título.

Dedico este trabalho aos meus pais, Valdevino e Elza, por sempre acreditarem em mim, me apoiarem em todas as minhas decisões e por terem se abdicado de suas vidas em prol das realizações e da felicidade de seus filhos.
E por minha irmã que sempre esteve ao meu lado me apoiando toda minha vida.
Nada disso teria sentido se vocês não existissem na minha vida.
Obrigado

RESUMO

A ossificação da arquitetura da Internet impõe dificuldade aos sistemas finais em recuperar informações de rede do caminho fim-a-fim, entre origem e destino. Particularmente, os emissores devem estimar as condições do enlace gargalo para ajustar suas taxas de transmissão. Este trabalho trata da investigação dos impactos no enlace de gargalo no desempenho de protocolos de transporte na Internet. O foco do trabalho está no estudo de método SBD (*Shared Bottleneck Detection*) em transmissões de múltiplos caminhos do protocolo MPQUIC (*MultiPath Quick UDP Internet Connection*). Essas conexões consistem em múltiplos subfluxos que percorrem diferentes caminhos fim-a-fim entre a origem e o destino, sendo consideradas o Estado da Arte no transporte de dados na Internet. Ao mesmo tempo, os mecanismos SBD são inovadores ao permitirem que os sistemas finais quantifiquem métricas estatísticas sobre a qualidade dos caminhos fim-a-fim (e.g., latências, perda de pacote) em uso pelos subfluxos. Com isso, o emissor infere se os gargalos dos caminhos são ou não compartilhados entre os subfluxos, para que se possa ajustar melhor a carga de trabalho nos subfluxos. A investigação é realizada experimentalmente através da implementação e validação do MPQUIC-SBD: uma extensão do MPQUIC que fornece suporte ao SBD de acordo com o Padrão da Internet descrito na RFC 8382. Resultados experimentais de emulações de transmissões DASH (*Dynamic Adaptive Streaming over HTTP*) mostram que MPQUIC-SBD é capaz de melhorar a Qualidade de Serviço e a Qualidade da Experiência em aplicações de vídeo.

ABSTRACT

The ossification of the Internet architecture imposes difficulty on the end-systems to retrieve network information of the end-to-end path, between source and destination. In particular, senders have to estimate the conditions of the bottleneck link in order to adjust their transmission rates. This work deals with the investigation of the impacts of bottleneck links on the performance of transport protocols on the Internet. The focus is on the study of SBD (Shared Bottleneck Detection) in multipath transmissions using the MPQUIC (MultiPath Quick UDP Internet Connection). Such UDP transmissions consist of multiple subflows that take different end-to-end paths between source and destination, being considered the State-of-The-Art of the data transport on the Internet. At the same time, SBD mechanisms are innovative when allowing the end-systems to quantify statistical metrics on the quality of end-to-end paths (e.g., latencies, packet loss) in use by subflows. Then, the sender infers whether the bottlenecks of the paths are or are not shared between subflows, so that one can better adjust the workload on the subflows. The investigation is carried out experimentally through the implementation and validation of MPQUIC-SBD: an extension of MPQUIC which provides SBD support according to the Internet Standard of RFC 8382. Experimental results from emulations of DASH (Dynamic Adaptive Streaming over HTTP) transmissions show that MPQUIC-SBD is capable of improving both Quality of Service and Quality of Experience in video streaming applications.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Bruno Kimura pela orientação e dedicação, que se não fosse ele, não poderia ter chegado até aqui.

Agradeço à Dra. Simone Ferlin (Ericsson AB, Suécia) pela orientação e acompanhamento junto à implementação.

Agradeço às Profa Özgü Alay (University of Oslo, Noruega) e Anna Brunstrom (Karlstad University, Suécia) pela orientação e importante ajuda na interpretação dos resultados.

Agradeço à CAPES pela bolsa.

Agradeço à FAPESP (processo #2015/18808) pelos equipamentos de laboratório.

SUMÁRIO

1 INTRODUÇÃO 1
- 1.1 Caracterização do Problema 2
- 1.2 Objetivos, Contribuições e Resultados 2
- 1.3 Motivações 3
- 1.4 Organização do Texto 4

2 MEDIÇÕES EM REDES DE COMPUTADORES 5
- 2.1 Termologia 5
- 2.2 Medição Ativa 7
  - 2.2.1 Técnicas de Medição Ativa 8
  - 2.2.2 Ferramentas de Medição de Capacidade 9
  - 2.2.3 Ferramenta de Medição de Latência 14
- 2.3 Medição passiva 15
  - 2.3.1 Métodos de Medição Passiva 15
  - 2.3.2 Ferramentas de Medição Passiva 17
  - 2.3.3 Medição Passiva em Protocolos de Transporte 20
- 2.4 Considerações finais 21

3 CONEXÕES RÁPIDAS COM UDP 23
- 3.1 QUIC - *Quick UDP Internet Connection* 23
  - 3.1.1 Estabelecimento de Conexões 24
  - 3.1.2 Pacotes QUIC 26
- 3.2 MPQUIC - *Multipath QUIC* 28
  - 3.2.1 Caminhos no MPQUIC 28
  - 3.2.2 Mecanismos Internos 29
  - 3.2.3 Controle de Congestionamento Acoplado 29
  - 3.2.4 Retransmissão Rápida 30
- 3.3 Considerações Finais 31

4 MPQUIC-SBD: DETECÇÃO DE GARGALOS COMPARTILHADOS NO MPQUIC 33
- 4.1 Método SBD conforme RFC 8382 33
  - 4.1.1 Medições de OWD 33
  - 4.1.2 Extração de Métricas Estatísticas 34
  - 4.1.3 Agrupamento de Subfluxos 37
- 4.2 Implementação no MPQUIC 42
  - 4.2.1 Considerações de Projeto 43
  - 4.2.2 Novos Campos nos Quadros 44
  - 4.2.3 Principais Procedimentos no Receptor 45
  - 4.2.4 Controle de Congestionamento baseado em SBD no Emissor 47
- 4.3 Trabalhos Relacionados 48
- 4.4 Considerações Finais 50

5 RESULTADOS EXPERIMENTAIS 53
- 5.1 Soluções Avaliadas 53
- 5.2 Aplicação de Video 54
  - 5.2.1 Implementação da Aplicação 54
- 5.3 Cenários Emulados de Gargalos de Rede 56

5.3.1 Configuração dos Gargalos 56
5.3.2 Geração de Tráfego de Fundo 57
5.4 Acurácia do Algoritmo de Agrupamento SBD 57
5.4.1 Planejamento dos Experimentos 58
5.4.2 Acurácia Observada 58
5.5 Qualidade de Serviço no Transporte MPQUIC 59
5.5.1 Vazão na transmissão do vídeo 60
5.5.2 Latência da transmissão do vídeo 61
5.6 Qualidade de Experiência na Aplicação de Video 62
5.6.1 Taxas de bits dos segmentos requisitados 62
5.6.2 Frequência de troca de taxas de bits 63
5.6.3 Rebufferização durante a reprodução do vídeo 64
5.7 Considerações Finais 65
6 CONCLUSÕES 67

REFERÊNCIAS BIBLIOGRÁFICAS 69

LISTA DE FIGURAS

Figura 1 Diagrama de fluxo *sankey* das técnicas, as principais ferramentas, as métricas e os protocolos da pilha utilizados em medições ativas. Fonte: do autor 10

Figura 2 Diagrama de fluxo *sankey* as técnicas, as principais ferramentas, as métricas e os protocolos da pilha utilizados em medições passivas. Fonte: do autor 18

Figura 3 QUIC em uma pilha HTTPS. Fonte: Imagem adaptada de [76]. 23

Figura 4 Estabelecimento conexão segura sem ter ocorrido comunicação prévia entre cliente e servidor. Fonte: Imagem adaptada de [103]. 25

Figura 5 Estabelecimento de uma nova conexão segura após já ter ocorrido comunicação prévia entre cliente e servidor. Fonte: Imagem adaptada de [103]. 26

Figura 6 Estrutura de um pacote QUIC. Em vermelho destaca-se o cabeçalho público autenticado, mas não criptografado. Em verde indica-se a carga útil que é criptografada. Essa estrutura de pacotes vem sendo evoluída à medida em que o protocolo QUIC passa por padronizações junto ao IETF. Fonte: imagem adaptada de [125, 55]. 26

Figura 7 Comparação dos fluxos QUIC, MPTCP e MPQUIC. Fonte: imagem adaptada de Viernickel *et al.* [125]. 28

Figura 8 Exemplo de uma conexão MPQUIC sobre dois caminhos. Fonte: imagem adaptada de [77]. 29

Figura 9 Diagrama do algoritmo de agrupamento Fonte: imagem adaptada de [46]. 38

Figura 10 Exemplo de um agrupamento na etapa 2, considerando uma conexão que contenha 8 subfluxos. Fonte: do autor 41

Figura 11 Componentes de uma implementação do MPQUIC e sua modificação (cinza) com o mecanismo SBD, considerando a abordagem de implementação mais complexa no receptor. Fonte: imagem adaptada de Viernickel *et al.* [125] 44

Figura 12 Emissor, receptor e seus quadros de fluxo de dados (*stream*) e reconhecimento (*ACK*) transmitidos em um subfluxo. Novo campos incluídos pelo mecanismo SBD são `PathID`, `LossCount`, `TimeStamp`, `Epoch` e `Groups` nos quadros de dados e de reconhecimento, respectivamente. Comprimento dos campos em bits. Fonte: do autor. 44

Figura 13 Fluxograma da operação do mecanismo SBD junto ao MPQUIC. Fonte: do autor. 45

Figura 14 Diagrama de Classes da implementação MPQUIC-SBD. Fonte: do autor. 46

Figura 15 Pilha de protocolos e implementação das soluções avaliadas nas camadas DASH e MPQUIC. Legenda: U (espaço de usuário), N (espaço de núcleo), SW (software), HW (hardware), NIC (*network interface controller*). 53
Figura 16 Cenário de gargalos compartilhados SB (*Shared Bottleneck*). 56
Figura 17 Cenário de gargalos não compartilhados (*Non-Shared Bottleneck*). 56
Figura 18 Acurácia do algoritmo de agrupamento SBD em cenário de gargalos não compartilhados (NSB). 59
Figura 19 Vazão da transmissão dos segmentos de vídeo. 60
Figura 20 Latência de transmissão dos segmentos de vídeo. 61
Figura 21 Qualidade média (*bitrate*) dos segmentos de vídeo requisitados. 63
Figura 22 Frequência de troca de qualidade de segmentos. 64
Figura 23 Frequência de rebufferização durante a reprodução do vídeo. 65

LISTA DE TABELAS

Tabela 1 Principal terminologia utilizada no texto. 5
Tabela 2 Principais ferramentas de medição ativa. 11
Tabela 3 Principais ferramentas de medição passiva. 18
Tabela 4 Parâmetros de tempo com valores recomendados pela RFC 8382 [46]. 34
Tabela 5 Limiares de utilizados pelo mecanismo de agrupamento de subfluxos previstos RFC 8382 [46]. 38
Tabela 6 Comparação entre os principais trabalhos relacionados. 50
Tabela 7 Soluções MPQUIC utilizadas avaliadas nos experimentos. 54
Tabela 8 Algoritmos ABR/DASH utilizados nos experimentos. 55
Tabela 9 Planejamento dos experimentos de acordo com diferentes configurações de tráfego de fundo (em números de fluxos). 58

1 INTRODUÇÃO

A Internet se caracteriza por ser uma rede complexa, onde uma gigantesca carga de trabalho é gerada por tráfego de pacotes de dados entre bilhões de sistemas finais, e.g., dispositivos de usuários e servidores. Para tratar esse tráfego, sempre crescente, inovação tecnológica tem ocorrido nos enlaces de dados das redes de acesso. Ao longo dos anos, pôde-se observar a evolução dos padrões da família IEEE 802.3 (ethernet) e IEEE 802.11 (WiFi), a chegada de FTTH (*Fiber-to-the-Home*) à redes domésticas, a evolução das redes celulares ao atual 4G, e as futuras redes 5G com implantação próxima. Essas tecnologias buscam melhorar a Qualidade de Serviços (QoS) para os usuários, essencialmente, elevando as capacidades dos enlaces de dados nas redes de acesso.

Considerando as camadas da pilha de protocolos TCP/IP, inovação tecnológica também aparece na camada de transporte e, de forma mais significativa, na camada de aplicação. No transporte de dados na Internet, observa-se a evolução do protocolo TCP para o *MultiPath* TCP (MPTCP) [34], de modo a possibilitar transmissões por múltiplos caminhos na Internet, explorando a diversidade de redes de acesso disponíveis nos sistemas finais *multihomed*. Ao agregar capacidades dos caminhos fim-a-fim disponíveis, o protocolo MPTCP possibilita maior taxa de transmissão e tolerância a falhas de conexões entre os sistemas finais envolvidos [35].

Na camada de aplicação, uma das principais inovações de comunicação dos últimos anos tem sido através das conexões rápidas UDP através do protocolo QUIC (*Quick UDP Internet Connections*) [55], originalmente proposto para possibilitar menor latência principalmente em transmissões HTTPS [76]. No protocolo QUIC há serviços de transmissão confiável equivalente aos disponíveis no TCP, como controle de congestionamento e controle de fluxo. Entretanto, o próprio protocolo já provê transmissão segura, com rápido estabelecimento de conexão (zero-RTT), com multiplexação de fluxos de dados dentro de uma mesma conexão. Além disso, diferente dos protocolos TCP e MPTCP, cuja implementação ocorre no núcleo dos sistemas operacionais, o protocolo QUIC é implementado no espaço de usuário, como componente de software junto à aplicação, facilitando a sua implantação, atualização e, consequentemente, a sua adoção. Recentemente, o protocolo QUIC tem recebido bastante atenção da academia e da indústria, havendo diversas implementações em várias linguagens [99]. Uma proposta evolui o protocolo QUIC em linguagem Go (`quic-go`) [116] para o *MultiPath* QUIC (MPQUIC) [20, 21], habilitando transmissões por múltiplos caminhos a partir de subfluxos QUIC/UDP. Assim como MPTCP, o MPQUIC possibilita melhor uso dos recursos de rede ao agregar capacidades dos múltiplos caminhos disponíveis e, com isso, aumentar a vazão fim-a-fim e reduzir ainda mais latência.

Embora tenham ocorrido melhorias nas tecnologias de transmissão e em protocolos de camadas superiores da pilha TCP/IP, a arquitetura da Internet não passou pela mesma inovação em seu projeto. Essa arquitetura, vislumbrada há mais de quatro décadas, está sujeita ao problema de enrijecimento de tecnologia, i.e., a ossificação [98].

Particularmente, a pouca flexibilidade dificulta os sistemas finais obterem informações sobre o tráfego corrente nos caminhos de dados. Especificamente, a arquitetura separa a operação da pilha de protocolos TCP/IP em dois segmentos: uma pilha mais complexa nos sistemas finais (e.g., dispositivos de usuários, servidores), outra pilha mais enxuta nos equipamentos de interconexão (e.g., roteadores, comutadores). Essa separação prevê que os sistemas finais nas redes de acesso tenham visibilidade apenas entre si, através de um caminho de dados fim-a-fim definido pelos pares de endereços IP e portas de origem e destino. Dessa forma, os sistemas finais não recuperam informações quanto ao estado dos enlaces que compõem os caminhos fim-a-fim, especialmente as condições atuais dos enlaces de gargalos. Para contornar esse problema, os protocolos de transporte, como TCP, MPTCP, QUIC e MPQUIC, os quais tomam decisões de controle de congestionamento e escalonamento de pacotes, devem inferir as condições dos caminhos medições passivas do tráfego fim-a-fim sobre a própria carga gerada pelas aplicações. Embora soluções de rede possam ser aplicadas para recuperar informações dos enlaces, como ECN (*Explicit Congestion Notification*) [105] e SDN (*Software-Defined Networks*) [71], essas soluções são custosas ao exigirem modificações na infraestrutura de rede e introdução de novos elementos.

1.1 CARACTERIZAÇÃO DO PROBLEMA

Em uma conexão de múltiplos caminhos, os subfluxos pertencentes a conexão pode compartilhar gargalos em comum e, com isso, disputarem recursos de redes entre si, levando a um problema conhecido como *contenção inter-caminhos* [70]. Sem saber se os subfluxos compartilham ou não gargalos entre si, a dosagem da carga de trabalho sobre os subfluxos pode ser equivocadamente definida pelos mecanismos de controle de congestionamento e escalonamento de pacotes. Umas das formas de se mitigar esse problema é prover, junto aos protocolos de transporte, o suporte à detecção de compartilhamento de gargalos, comumente denominado mecanismo SBD (*Shared Bottleneck Detection*).

Conforme o padrão de Internet definido pela RFC 8382 [46], um mecanismo SBD prevê o uso de métodos de medições passivas, principalmente de latência, sobre cada subfluxo existente em uma conexão de múltiplos caminhos, para a extração de métricas estatísticas. Algoritmos de agrupamento podem então ser aplicados para determinar os grupos de subfluxos que compartilham gargalos a partir das similaridades das métricas observadas pelos subfluxos. Ao passo que os benefícios dos mecanismos SBDs em transmissões do protocolo MPTCP são conhecidos [30, 31, 128], poucos se sabe sobre sua aplicação em transmissões do protocolo MPQUIC.

1.2 OBJETIVOS, CONTRIBUIÇÕES E RESULTADOS

Este trabalho investiga os impactos no enlace de gargalo no desempenho de protocolos de transporte na Internet. O objetivo está no estudo de método SBD em transmissões de múltiplos caminhos do protocolo MPQUIC.

As principais contribuições deste trabalho estão na proposta, implementação e validação do protocolo **MPQUIC-SBD**. No presente momento, MPQUIC-SBD é a primeiro protocolo que estende MPQUIC para fornecer suporte SBD de acordo com o Padrão da Internet descrito na RFC 8382 [46].

Avaliações de desempenho foram realizadas a partir de transmissões de aplicações de vídeo. Essas aplicações são de interesse, uma vez que suas transmissões representam o tráfego majoritário existente na Internet atualmente, superior a 75% de todo o tráfego IP [16] e em média 60% do tráfego de *smartphones* [28].

Resultados experimentais a partir de emulações de transmissões DASH (*Dynamic Adaptive Streaming over HTTP*) – o padrão de vídeo *streaming* na Internet [118] mostram que MPQUIC-SBD é capaz de melhorar parâmetros QoS para aplicações de vídeo. Mesmo com acurácia mais modestas ($\approx$ 60%) na detecção de compartilhamento de gargalos, MPQUIC-SBD possibilitou ganhos de 25% na vazão de transmissão dos segmentos de vídeo quando comparado à versão padrão do protocolo MPQUIC. Com isso, melhores indicadores de Qualidade de Experiência (QoE) puderam ser observados com diferentes algoritmos ABR (*Adaptive Bit Rate*) na aplicação de vídeo.

1.3 MOTIVAÇÕES

A densidade de dispositivos conectados na Internet tem levado a um tráfego de dados historicamente crescente na rede. Por essa razão, há uma incessante busca por melhor desempenho de transmissões, em diferentes camadas da pilha TCP/IP. Ao mesmo tempo, devido à ossificação da pilha de protocolo da Internet e, consequentemente, a ofuscação do núcleo da Internet para os sistemas finais, é um desafio identificar as causas de degradação de desempenho nos enlaces que compõem o caminho fim-a-fim. Um dos principais fatores que degradam o desempenho é o congestionamento no elance de gargalo.

Com o avanço das ferramentas de medição de tráfego na Internet, tem-se conseguido identificar o ponto onde ocorre o gargalo com mais precisão. Estudos mostraram que 60% da perda de pacote está a dois saltos de distância da origem, e que menos de 10% dos computadores em um *cluster* compartilham o gargalo por mais da metade do tempo [51]. Outros estudos, que identificam gargalos em redes domésticas, mostram ser mais comum a ocorrência de um congestionamento no gargalo em redes sem fio que em enlaces de acesso [122]. Nesse contexto, surge o problema de como mensurar a degradação dos enlaces no caminho fim-a-fim, algo que não é nada trivial. Para tanto, tem-se normalmente utilizado estimativas, modelos e novos algoritmos para obter maior precisão quanto às condições dos gargalos [30, 72].

Buscando melhorias, diferentes soluções foram propostas para diminuir o atraso do fluxo de pacotes [74, 13, 33, 92, 132, 30, 113]. Alguns autores tratam desse tema diretamente na camada de aplicação. Em Carter e Crovella [13], os autores aprimoraram o protocolo FTP (*File Transfer Protocol*), estimando tanto a largura de banda disponível quanto o congestionamento do gargalo. Em transmissões por múltiplos caminhos, novos algoritmos têm sido criados, possibilitando mais precisão na identificação do compartilhamento do gargalo e, consequentemente, obtenção de ganho na taxa de transferência [30, 31]. Também há soluções mais custosas apoiadas em modificações de infraestrutura de rede. Pode-se destacar, recentemente, os novos algoritmos [129] de escalonamento de pacotes e de controle de congestionamento acoplado com suporte a detecção de compartilhamento de gargalo baseado em ECN. Além de ECN, suporte de redes SDN tem sido utilizado para evitar que os subfluxos pertencentes a uma conexão sempre utilizem um mesmo caminho, possibilitando ganhos significativos na taxa de transmissão [113]. Embora esforços tenham sido fei-

tos no aprimoramento de transmissões por múltiplos caminhos, esses esforços sido direcionados ao protocolo MPTCP.

O protocolo MPQUIC [20], por outro lado, é uma solução de transporte *multi-path* recente, com poucas implementações disponíveis [21, 125], e, consequentemente, ainda pouco explorado quando comparado ao protocolo MPTCP. Isso possibilita oportunidades de investigações de problemas em aberto em transmissões do MP-QUIC, especialmente de temas mais complexos, como a detecção de gargalos compartilhados entre subfluxos QUIC/UDP através de métodos não invasivos baseados em medições passivas, sendo o foco deste trabalho.

1.4 ORGANIZAÇÃO DO TEXTO

O restante deste trabalho está organizado da seguinte forma. O próximo capítulo apresenta a visão geral das principais técnicas e métodos de medição ativa e passiva, bem como uma revisão da terminologia utilizada. O Capítulo 3 descreve o protocolo de conexões rápidas com UDP na Internet, apresentando as principais operações dos protocolos QUIC e MPQUIC. O Capítulo 4 apresenta a principal contribuição deste trabalho de mestrado, discutindo o desenvolvimento do protocolo MPQUIC-SBD. O Capítulo 5 discute os resultados experimentais de transmissão de aplicações de vídeo em ambiente emulado. Por fim, o Capítulo 6 apresenta as conclusões deste trabalho e direcionamentos para trabalhos futuros.

2 MEDIÇÕES EM REDES DE COMPUTADORES

Neste capítulo são apresentadas as principais técnicas e ferramentas de medições em redes de computadores. A seção seguinte apresenta os principais termos usados. As técnicas e ferramentes de medição ativa e passiva são discutidas nas Seções 2.2 e 2.3. Por fim, a Seção 2.4 traz as considerações finais deste capítulo.

2.1 TERMOLOGIA

Os principais termos empregados neste trabalho estão listados e categorizados na Tabela 1, bem como definidos ao longo desta seção. As categorias envolvem a terminologia relacionada a definições conceituais sobre Infraestrutura de rede, Transporte de Dados e Métricas de Aferição de Transmissão.

Tabela 1: Principal terminologia utilizada no texto.

Termo	**Categorias**		
	Infraestrutura de Rede	Transporte de Dados	Métrica de Aferição
Nó (*Node*)	X		
Enlace (*Link*)	X		
Caminho (*Path*)	X		
Fluxo (*Flow*)		X	
Tráfego (*Traffic*)		X	
Capacidade			X
Largura de Banda (*Bandwidth*)			X
LBD (Larg. de Banda Disponível)			X
Tx de Transf. Efetiva (*Throughput*)			X
Tx de Transf. dos Dados Úteis (*Goodput*)			X
BTC (*Bulk Transfer Capacity*)			X
Latência (*Delay*)			X
Jitter			X
Tx de Pacotes Perdidos (*Packet Loss Rate*)			X
RTT - Tempo de Ida e Volta			X
Tempo de Ida (*One-Way Delay - OWD*)			X
TE - TCP emulado (*Emulated TCP*)			X

O conceito de **métricas (*Metrics*)** se refere à um dado quantitativo relacionado ao desempenho e a confiabilidade da rede, o qual é crucial para caracterizar o estado atual do tráfego num caminho. Uma única medida de uma métrica pode ser chamado de métrica *singleton*, um conjunto de medidas distintas de *singletons* pode ser chamado de métrica de amostra, bem como uma estatística calculada sobre uma métrica de amostra é chamado de métrica estatística [97].

Um **Nó (*Node*)** é um computador capaz de executar serviços, programas de usuários, comunicar-se usando a internet e protocolos. Alguns nós são chamados de cli-

entes de redes de computadores, enquanto outros implementam serviços de encaminhamento [112, 97].

O **Enlace (*Link*)** é uma conexão entre dois nós de dispositivo de redes de computadores, o qual transmite dados de um nó ao outro [97].

Um **Caminho (*Path*)** ou caminho fim-a-fim é uma série de elances conectados em uma sequência de nós de encaminhamento, os quais estão entre o nó sistema final de origem e o nó sistema final de destino, sendo n o número de enlaces, um caminho deve ter $n \geqslant 1$ enlaces [15].

Um **Fluxo (*Flow*)** é um conjunto de pacote que passa por um caminho em um intervalo de tempo [6, 18]. O fluxo pode ser caracterizado por direção, simetria, caminho, número de pacotes, número de bytes, endereços de origem e destino [11].

O **Tráfego (*Traffic*)** é um conjunto de fluxo observado de um caminho em um intervalo de tempo [10].

A **Capacidade** de um elance é a taxa máxima de transferência que um elance pode fornecer para um fluxo, quando não houver tráfego neste elance. A capacidade de um caminho é limitada pela menor capacidade de um elance pertencente ao caminho [1].

A **Largura de Banda (*Bandwidth*)** refere à quantidade de dados que um elance ou caminho pode fornecer por unidade de tempo [102]. A unidade de medição é em bits por segundos (bps).

A **LBD (Largura de Banda Disponível) (*Available Bandwidth*)** em um elance é sua capacidade não utilizada em algum intervalo de tempo T. O elance pode estar ocioso ou transmitindo em sua capacidade máxima a qualquer instante em T, sua largura de banda disponível é aquela naquele instante. Estendendo o conceito para caminho P, a largura de banda disponível ao longo de P é a mínima disponível dentre todos os elances que compreendem P [121]. Portanto, largura de banda disponível é taxa máxima de transferência que um caminho pode oferecer a um fluxo, sem reduzir a taxa de transferência do restante do tráfego em P.

A **Taxa de Transferência Efetiva (*Throughput*)** é uma medida da quantidade de dados transferidos por um elance em um determinado período de tempo. A taxa de transferência é comumente medida em bits por segundo, podendo também ser observada em bytes [90].

A **Taxa de Transferência dos Dados Úteis (*Goodput*):** é um subconjunto da taxa de transferência efetiva, que consiste em tráfego útil. Ou seja, exclui pacotes duplicados, pacotes que serão descartados no fluxo, fragmentos de pacotes que são descartados no receptor por não poderem ser remontados em pacotes completos e assim por diante [32]. Comumente, o *goodput* é uma métrica associada a taxa útil de dados observada nas camadas de transporte e/ou aplicação.

O **BTC (*Bulk Transfer Capacity*)** é definido como a taxa de transferência de dados que um dispositivo de rede pode transmitir usando o controle de congestionamento padrão através de um determinado caminho de rede [85].

A **Latência (*Delay*):** se refere ao atraso nodal composto pela soma de diferentes tipos de atrasos, como atraso no processamento, atraso na propagação, atraso de enfileiramento e atraso de transmissão. Atraso no processamento é tempo que um nó no caminho leva para processar o pacote. Isso inclui, por exemplo, o tempo do dispositivo de rede leva para verificar erros de bits no quadro, examinar o cabeçalho do pacote, determinar para onde deve encaminhá-lo. O atraso de transmissão é a quantidade de tempo necessária para um dispositivo de rede colocar todos os bits

do pacote no enlace de saída. O atraso de propagação é o tempo necessário para o sinal (um único bit) percorrer de uma extremidade a outra do meio de transmissão. Esse atraso depende do meio físico, sendo determinado pela distância entre dois nós divididos pela velocidade de propagação do bit. A atraso de enfileiramento é o tempo que um pacote passa na fila de um nó roteador, aguardando a sua transmissão. O atraso na fila é proporcional ao tamanho do *buffer* e à quantidade de tráfego cruzado que entra no roteador. Neste trabalho, a noção de atraso é associada ao atraso fim-a-fim, portanto, a soma dos atrasos nodais de cada roteador existente no caminho.

O **Jitter** é o valor absoluto da diferença entre o atraso de encaminhamento de dois pacotes recebidos consecutivos pertencentes ao mesmo fluxo. O *jitter* pode ser expresso como $|D(i) - D(i-1)|$, onde D é igual ao atraso de encaminhamento e i é a ordem em que os pacotes foram recebidos [100].

A **Taxa de Pacotes Perdidos (*Packet Loss Rate*)** é a métrica que quantifica a taxa média de perda de pacotes ocorrida no fluxo [44].

O **Tempo de Ida e Volta (*Round-Trip Time - RTT*)** foi originalmente definido como métrica interna do protocolo TCP, sendo tempo que leva para um pacote chegar ao destino e voltar a origem em uma transmissão fim-a-fim. O RTT também é conhecido como atraso bidirecional.

O **Tempo de Ida (*One-Way Delay - OWD*)** é o atraso fim-a-fim definido anteriormente, medindo o tempo que um pacote leva para ser transmitido da origem ao destino. O OWD é também conhecido como atraso unidirecional RFC2680 [4].

Uma conexão **TE - TCP emulado *(Emulated TCP)*** é uma conexão TCP sintética entre dois nós na rede, que em muitos casos é utilizada para aferir métricas de rede, como medidas de latência, capacidade, etc.

2.2 MEDIÇÃO ATIVA

As medições ativas introduzem pacotes sintéticos na rede, denominados sondas, para, por exemplo, inferir o tempo que leva para o pacote chegar à outra extremidade da rede (atraso unidirecional), aferir a largura de banda disponível ponta a ponta ou atraso bidirecional. A ideia por trás desse tipo de medição é, por amostragem, chegar a conclusões sobre a qualidade de uma determinada parte da rede.

As ferramentas de medições ativas normalmente não exigem grandes quantidades de espaço de armazenamento, sendo úteis na obtenção de métricas que não são possíveis de ser obtidas através de medições passivas. Por exemplo, ao usar a sondagem ativa, não há problemas de privacidade, pois os dados usados não contêm nenhuma informação privada de usuários e suas aplicações. Os pacotes possuem carga útil artificial, sendo gerados sob demanda e, portanto, contendo apenas bits aleatórios. Entretanto, ao introduzir esses pacotes na rede, causará uma pertubação no fluxo normal do tráfego. Nesse caso, o fluxo de pacotes de sondagem compete pelo uso de recursos de rede com o fluxo de pacotes originados pelas aplicações dos usuários dessa rede. Portanto, diferentes condições de tráfego podem acarretar incertezas na medição. A medição passiva não ocorre esse problema, uma vez que os protocolos realizam as medições sobre o próprio fluxo de pacotes das aplicações dos usuários, não introduzindo nenhum tráfego sintético de sondagem é introduzido na rede.

As ferramentas de medição ativas mais conhecidas são provavelmente o `traceroute` e o `ping`, sendo integrados à maioria dos sistemas operacionais atualmente. Ambas ferramentas são baseadas no envio de sondas por pacotes ICMP.

2.2.1 *Técnicas de Medição Ativa*

Tanto para o bom gerenciamento e operação de rede quanto para a eficiência de protocolos de rede, conhecer as métricas de tráfego corrente é fundamental. Por exemplo, conhecer a latência dos elances/caminhos, número de pacotes perdidos, a largura de banda disponível, a capacidade do elance/caminho. Nesse contexto, serviços de gerenciamento ou protocolos de rede se beneficiam dessas estimativas, uma vez que podem controlar o tráfego, adaptar taxa de envio do emissor de forma e compartilhar recursos de rede de forma mais eficiente e justa. Para tanto, diferentes métodos de medição ativa podem ser aplicados.

No caso da medição de latência, o método mais empregado é de Requisição e Resposta (RR), medindo o tempo de ida e volta de um pacote, seja por sondas ICMP, datagramas UDP ou segmentos TCP. Para a medição de capacidade, há quatro principais técnicas para estimar a capacidade e largura de banda disponível [102]. Sejam elas VPS (*Variable packet size*), PPTD (*Packet Pair/Train Dispersion*), SLoPS (*Self-Loading Periodic Streams*) e TTOP *Trains of Packet Pairs*).

A técnica **VPS** busca estimar a capacidade de cada elance ao longo de um caminho, sendo Bellovin [7] e Jacobson [58] os pioneiros na sua proposição e exploração. Tal técnica prevê o envio de pacotes de tamanhos diferentes do nó de origem para todos os roteadores que está entre o nó de origem e o nó de destino. O tempo de ida e volta (RTT) de cada pacote é medido, de modo que a capacidade do elance pode ser estimada a partir da razão do tamanho do pacote pelo tempo RTT observado. Trabalhos subsequentes foram propostos para melhor a técnica [26, 73, 95].

A técnica **PPTD** é usada para estimar a capacidade de um caminho. O nó de origem envia um par de pacotes para o nó de destino. Os pacotes no par possuem o mesmo tamanho e a dispersão ao longo do caminho é medida no nó de destino. Se houver um enlace de gargalo no caminho a dispersão aumentará. Embora a técnica seja interessante, ela pressupõe que o tráfego no caminho se aproxime de zero [102]. Para contorna esse problema, vários pares de pacote são enviados, de modo a diminuir o ruído de medição e aumentar a chance de que um par de pacote possa ser recebido sem interferência de outro fluxo durante a sua transmissão. A técnica PPTD são discutidas em diversos trabalhos [56, 67, 8].

A técnica **SLoPS** prevê a estimativa da largura de banda disponível de um caminho [61]. O nó de origem envia uma sequência de pacotes de tamanhos iguais a uma taxa crescente e monitora as variações de atrasos unidirecionais. O nó de destino notifica o nó de origem dos atrasos monitorados. Para evitar sobrecarga na rede, cria-se períodos ociosos entre fluxos sucessivos. Se o fluxo gerado pelo envio dos pacotes de sondagem for maior que a largura de banda disponível, impactará no aumento do atraso, indicando congestionamento num possível elance de gargalo no caminho. Utilizando uma abordagem similar à busca binária, SLoPS tenta aproximar a taxa de envio de pacotes à largura de banda disponível.

Outra técnica, denominada **TOPP**, para medir a disponibilidade da largura de banda foi proposta por Maier *et al.* [81]. A técnica é semelhante ao SLOPS, porém, a

diferença é que, além de estimar a banda disponível, TOPP também estima a capacidade do elance do gargalo. Para tanto, a técnica usa sequências de pares de pacotes enviados do nó de origem para o nó de destino a taxas crescentes. A partir da relação entre as taxas de entrada e saída de diferentes pares de pacotes, pode-se estimar a disponibilidade de banda e a capacidade do elance no caminho. Análise comparativa de ferramentas e metodologias de medição de largura de banda podem ser encontradas em Ali *et al.* [1].

Todas essas técnicas são intrusivas pela própria natureza da medição ativa [102] ao depender da introdução de pacotes de sondagem na rede. O impacto do uso pode consumir recursos de rede estimado em até 10% da largura de banda disponível [102]. Outra desvantagem é que essas técnicas assumem a premissa de que o caminho e o tráfego medido permanecem estáticos. Entretanto, alterações dinâmicas que ocorrem tanto no tráfego quanto no roteamento podem causar ruídos nas medições, influenciando os resultados das métricas [135]. Nas próximas seções são apresentadas as principais ferramentas que aplicam essas técnicas nas medições de capacidade e latência.

2.2.2 *Ferramentas de Medição de Capacidade*

Há diversas ferramentas para medição ativa. A Figura 1 apresenta o diagrama de fluxo do tipo *sankey*, considerando os métodos, as principais ferramentas, as métricas e os protocolos da pilha utilizados em para realização medição ativa. Tabela 2 descreve essas ferramentas, as quais são detalhadas a seguir.

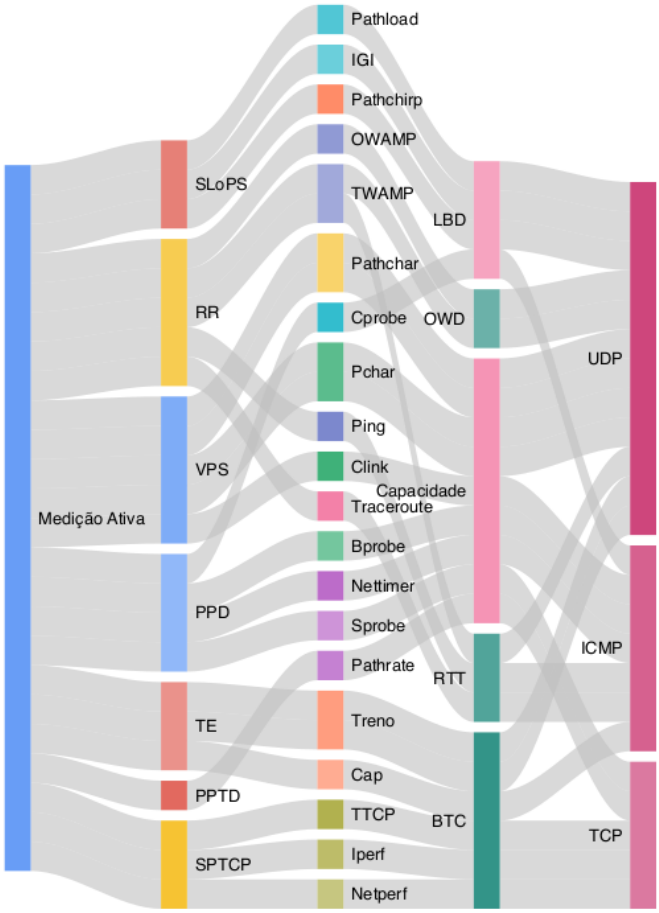


Figura 1: Diagrama de fluxo *sankey* das técnicas, as principais ferramentas, as métricas e os protocolos da pilha utilizados em medições ativas.
Fonte: do autor

Tabela 2: Principais ferramentas de medição ativa.

Ref.	Ferramenta	Descrição
[58]	Pathchar	Ferramenta que infere as características dos elance ao longo de um caminho.
[25]	Clink	Similar ao Pathchar, com a diferença que faz uso da técnica de "ímpar-par".
[83]	Pchar	Faz uso de regressão linear para estimar a largura de banda.
[12]	Bprobe	Usa a dispersão de pares de pacotes para estimar a capacidade de um caminho.
[74]	Nettimer	Usa técnica baseda em estimativa de densidade de *kernel* para processar medições de pares de pacotes e estimar a capacidade de um caminho.
[24]	Pathrate	Usa a técnica de pares de pacotes e trens de pacote fazendo uso de inúmeros pares de pacotes de tamanho variável para estimar capacidade de um caminho.
[114]	Sprobe	Ferramenta que fornece uma estimativa rápida da capacidade. A ferramenta é executada apenas no nó de origem.
[12]	Cprobe	Ferramenta pioneira para estimar a largura de banda disponível, faz uso de dispersão de um trem de oito pacotes para estimativa.
[60]	Pathload	Relata um intervalo em vez de uma única medição, estimando a variação da largura de banda disponível durante as medições.
[91]	IGI	Semelhante ao Pathload, busca encontrar a menor latência de medição.
[109]	Pathchirp	Semelhante ao Pathload, busca uma menor latência de medição.
[84]	Treno	Foi a primeira ferramenta proposta para medir o BTC. Treno simula uma conexão TCP enviando pacotes UDP.
[2]	Cap	Mesma finalidade que Treno, porém, com maior precisão e requer acesso às duas extremidades do caminho.
[124]	TTCP	Ferramente de *benchmark*, realiza teste no protocolo TCP enviando pacotes o mais rápido possível.
[123]	Iperf	Semelhante ao TTCP, porém adiciona recurso de múltiplas conexões TCP na medição.
[62]	Netperf	Semelhante ao TTCP, porém adiciona recurso de múltiplas conexões TCP na medição.
[130]	Ping	Mede conectividade, tempo de ida e volta e taxa de perda de pacotes entre dois nós finais.
[133]	OWAMP	Mede o atraso unidirecional em alta precisão.
[48]	TWAMP	Semelhante ao OWAMP, medindo também o atraso bidirecional.
[130]	Traceroute	Resolve o caminho entre dois nós e retorna uma única amostra de tempo de ida e volta para cada salto presente no caminho.

2.2.2.1 *Estimativa de capacidade dos elances*

A característica comum das ferramentas que realizam para estimar a capacidade de cada elance no caminho é o uso da técnica de sondagem VPS.

Pathchar é uma ferramenta que infere as características dos elance ao longo de um caminho, como latência, largura de banda, atrasos na fila. Foi a primeira ferramenta a implementar a detecção por VPS. Esta ferramenta foi escrita por Jacobson [58] e lançada em 1997.

Clink (*Characterize Links*) é um ferramente de código fonte aberta que faz o mesmo que o pathchar, ou seja, estima a latência e a largura de banda dos elances do caminho através do envio de pacotes UDP do nó de origem ao destino e medindo os tempos de ida e volta. A diferença é que Clink usa uma técnica "ímpar-par" [26] para gerar estimativas de capacidade de intervalo. O mecanismo é semelhante ao ping e ao traceroute, exceto pela necessidade de enviar uma quantidade de pacotes muito maior [25].

Pchar é outra implementação de código aberto que faz uso da técnica de VPS [83]. A biblioteca de captura de pacotes Libpcap é usada para obter registros de data e

hora do sistema operacional. Pchar provê três algoritmos de regressão linear para obter a inclinação das medidas mínimas de RTT, em relação ao tamanho do pacote de sondagem [102]·

2.2.2.2 *Estimativa de capacidade do caminho*

Essas ferramentas tentam estimar a capacidade do elance mais estreito ao longo de um caminho de ponta a ponta. A maioria delas utiliza a técnica de dispersão de pares de pacotes.

A ferramenta **Bprobe** [12] usa a dispersão de pares de pacotes para estimar a capacidade de um caminho. O Bprobe processa as medições de pares de pacotes com uma técnica de filtragem de união e interseção, na tentativa de descartar as medições de pares de pacotes afetadas pelo tráfego cruzado. Além disso, Bprobe usa pacotes de análise de tamanho variável para melhorar a precisão da ferramenta quando os pacotes de tráfego cruzado têm alguns tamanhos fixos, por exemplo, 40, 576 ou 1500 bytes. Bprobe exige execução apenas no nó de origem, uma vez que o nó de destino responde através com pacotes ICMP, protocolo o qual é nativo nos diversos sistemas operacionais existentes. Eventualmente, as respostas ICMP podem ser limitadas para evitar ataques de negação de serviço, o que afeta negativamente a precisão da medição [102]. Outro ponto a destacar é que essa ferramenta conta com o recurso do sistema operacional IRIX para hardware SGI (*Silicon Graphics*), um cronômetro de alta precisão que fornece um tempo de granularidade mais fino dos pacotes de sonda. Especificamente, enquanto a resolução normal do tempo de um relógio do sistema é de dezenas (ou mais) milissegundos, essas ferramentas são baseadas em um dispositivo mapeado pela memória SGI com uma resolução de 40 nanossegundos.

A ferramenta **Nettimer** pode ser executada de duas formas, por par de pacotes ou por sondagem VPS, embora esta última forma não descrita em sua documentação. Nettimer ter por objetivo estimar a capacidade do caminho. Para isso, a ferramenta usa uma técnica estatística sofisticada ao aplicar estimativa de densidade de *kernel* para processar medições de pares de pacotes. Um estimador de densidade de *kernel* identifica o modo dominante na distribuição de medições de pares de pacotes sem assumir uma certa origem para a distribuição de largura de banda, superando a limitação correspondente das técnicas baseadas em histograma [102]. O erro com essa técnica é inferior a 10%, e no pior caso de 40% [74].

Pathrate [23] é uma ferramenta capaz de estimar a capacidade do caminho, baseada na dispersão de pares e trens de pacotes. Para contornar o problema da multimodalidade da distribuição de dispersão de pares, Pathrate usa inúmeros pares de pacotes de tamanhos variáveis. A sua análise da distribuição das medidas resultantes revela todos os modos locais, estando um deles normalmente relacionados à capacidade do caminho. Em seguida, a ferramenta envia longos trens de pacotes para estimar a taxa média de dispersão ADR (*Asymptotic Dispersion Rate*) do caminho. A capacidade do caminho é sempre superior ao ADR, portanto, ADR irá fornecer um resultado inferior à capacidade. Pathrate foi projetada para ser robusta aos efeitos de tráfego cruzado, o que significa que ela pode medir a capacidade do caminho, mesmo quando o caminho estiver carregado significativamente. No entanto, a falta de escalabilidade e a necessidade de acessar os dois nós finais do caminho, tornam Pathrate não adequado para um estudo em larga escala das características dos caminhos da Internet [27].

Sprobe é uma ferramenta que fornece uma estimativa rápida da capacidade, sendo executada apenas no nó de origem do caminho. A ferramenta requer um conjunto de pacotes TCP SYN, totalizando 40 bytes, no meio dos quais um par grande de pacotes (1500 bytes) é inserido colocado. O nó de destino responde com um trem de mesmo tamanho com pacotes de TCP RST. SProbe então analisa a dispersão do tempo dos pacotes no trem e usa heurísticas para determinar a presença de tráfego cruzado através de reordenação ou elances multicanais [114].

2.2.2.3 *Estimativa de largura de banda disponível*

Cprobe [12] é uma ferramenta pioneira para estimar a largura de banda disponível usando medições fim-a-fim. Em vez de usar um par de pacotes, Cprobe mede a dispersão de um trem de oito pacotes. No entanto, foi demonstrado anteriormente em [24, 87] que trens longos de pacotes medem a taxa de dispersão, que não é a mesma que a largura de banda disponível do caminho. Em geral, a taxa de dispersão depende de todos os elances no caminho, bem como da taxa inicial do trem. Por outro lado, a largura de banda disponível depende apenas do elance mais estreito do caminho. Como em Bprobe, Cprobe também usa a ferramenta SGI para ter maior preciso do tempo.

Pathload [60] implementa a técnica de SLoPS. A ferramenta requer acesso às duas extremidades do caminho, mas não requer privilégios de superusuário, enviando apenas pacotes UDP. Pathload relata um intervalo em vez de uma única estimativa. O centro desse intervalo é a largura de banda média disponível durante as medições, enquanto o próprio intervalo estima a variação da largura de banda disponível durante as medições. Mais recentemente, duas novas ferramentas foram propostas para a estimativa da largura de banda disponível: **IGI** [91] e **pathChirp** [109]. Essas ferramentas modificam a metodologia de carregamento automático de TOPP ou SLoPS, usando diferentes padrões de fluxo de pacotes de sondagem. O principal objetivo de IGI e pathChirp é obter uma precisão semelhante ao de Pathload, mas com menor latência de medição.

2.2.2.4 *Medição de taxa de transferência TCP e BTC*

Treno [84] foi a primeira ferramenta a medir o BTC de um caminho. Para tanto, simula-se uma conexão TCP enviando pacotes UDP para o nó de destino, forçando o receptor responder com mensagens ICMP. Desta forma, só é necessário execução no nó de origem. Assim como em bprobe, o fato de as respostas ICMP por ventura serem limitadas, pode afetar negativamente a precisão da medição.

Cap é uma ferramenta para medição do BTC. Semelhante à ferramenta Iperf e ao NetPerf, com a diferença que usa pacotes UDP para emular o TCP. E Demonstrou ter maior precisão que Treno para medição de BTC [2].

TTCP (Teste TCP) [124], **NetPerf** [62] e **Iperf** [123] são as ferramentas de *benchmark* que usam grandes transferências de pacotes TCP para medir a taxa de transmissão efetiva (*goodput*) fim-a-fim. Todas as ferramentas usam métodos de medição ativa semelhantes. Um cliente e um servidor são configurados nos dois sistemas finais que representam as extremidades do caminho. Várias amostras das vazões máximas alcançadas são medidas durante a transferência. Sendo que o usuário pode controlar tamanho do *buffer* e o tamanho máximo da janela do TCP. NetPerf e Iperf são

ferramentas mais recentes do que TTCP, sendo escrita em 1984, com melhorias no processo de medição, além de possibilitar o recurso de estabelecer várias conexões TCP simultâneas entre o cliente e o servidor.

Assim como **TTCP**, **Iperf** e **NetPerf** tentam consumir o máximo da largura de banda disponível durante a medição. Dessa forma, essas ferramentas são mais invasivas que aquelas baseadas em VPS e PPTD, por exemplo. No entanto, como às três ferramentas utilizam TCP, o tráfego de suas conexões são controlados pelo controle de congestionamento do nó de origem, possibilitando que outras aplicações de rede não sejam bloqueadas durante a medição. Embora essas ferramentas sejam consideradas mais invasivas no uso da largura de banda, essa ferramentas são mais amigáveis mediante ao congestionamento que VPS e PPTD [29].

2.2.3 *Ferramenta de Medição de Latência*

Nesta seção são apresentadas ferramentas de medição ativa para determinar métricas de latência.

2.2.3.1 *Medição Fim-a-Fim*

O bem-conhecido **Ping**, além medir disponibilidade do par remoto, mede o tempo de ida e volta, bem como a taxa de pacotes perdidos entre dois nós finais [130, 135]. A ferramenta funciona enviando sondas ao destinatário através de mensagem ICMP do tipo *echo request*. O nó de destino, por sua vez, caso esteja operando, responde com outra mensagem ICMP, mas do tipo *echo reply*. O nó requisitante que envia a medição geralmente envia várias mensagens ICMP, sequencialmente. Se uma ou mais respostas forem recebidas, é garantida a conectividade entre os dois nós fim-a-fim, pelo menos parcialmente. O tempo de ida e volta é então calculado com o RTT entre as mensagens de requisição e resposta ICMP. A taxa de perda de pacotes é estimada simplesmente pela relação entre respostas perdidas e total de requisições enviadas. Mas até mesmo o Ping tem seus problemas. Muitos provedores de Internet começaram a filtrar pacotes ICMP devido a um número crescente de *worms* na Internet, como *Welchia* que envia inúmeras requisições ICMP para descobrir novas localizações de dispositivos para serem infectados [49, 119]. Além disso, o próprio nó de destino ou seu *firewall* podem não responder solicitações ICMP com o objetivo de ocultar localização. Por mais que essas limitações diminuem a utilidade do Ping, ainda sim, o Ping é uma ferramenta muito simples e bastante útil, estando disponível na maioria dos sistemas operacionais.

OWAMP, definido na RFC 4656, é um protocolo de medição ativa de alta precisão das características unidirecionais de um caminho, como atraso unidirecional e perda unidirecional [133]. Para a medição do atraso na rede, um cliente e um servidor é configurado entre o nó de origem e destino. O cliente envia pacotes UDP ao servidor carimbados com marcação do tempo do envio. A data e hora, TTL e número de sequência são registrados no momento do recebimento do pacote. Ao comparar a marcação do tempo do envio com a do tempo de chegada, calcula-se o atraso que o pacote sofreu no caminho. Então, o servidor envia os resultados da medição para o cliente. Nesse contexto, é importante que os relógios de ambos cliente e servidor estejam sincronizados para que a medição tenha maior precisão. Para tanto, o protocolo

assume que os tempos dos nós sejam sincronizados por fontes precisas, como GPS ou estando diretamente, ou próximos de servidores de tempo NTP primários.

TWAMP é um protocolo de medição ativa bidirecional definido pela RFC 5357 [48] que amplia OWAMP ao adicionar de RTT. TWAMP configura o cliente e servidor em nós de origem e de destino, possibilitando a medição bidirecional. A ferramenta apresenta uma precisão de medição melhor para o tempo de ida e volta fornecido pelo Ping. Além disso, como as mensagens TWAMP são encapsuladas em datagramas UDP, não apresenta as limitações que o Ping com ICMP.

2.2.3.2 *Medição de Múltiplos Saltos*

A ferramenta **Traceroute** é capaz de identificar os nós presentes no caminho entre dois sistemas finais, retornando amostragens de tempo de ida e volta para cada salto no caminho [130]. *Traceroute* faz isso enviando mensagens UDP ao nó de destino. Para descobrir os saltos, o valor de TTL (*Time to Live*) no cabeçalho IP é iniciado em 1 e incrementado para atingir progressivamente os saltos até o destino. Os roteadores que encaminham o pacote diminuirá o valor do TTL em um. O roteador em que o valor do TTL chegar a zero então descarta o pacote retorna uma mensagem ICMP de TTL excedido ao nó de origem. Assim, à medida que as sondas avançam pelos saltos do caminho, o respectivo roteador do salto cujo TTL é zero enviará mensagens de erro ICMP ao nó de origem. O nó de origem, por sua vez, receberá as mensagens ICMP dos roteadores, possibilitando-o determinar o RTT até esse roteador, além identificá-lo pelo seu respectivo endereço IP.

2.3 MEDIÇÃO PASSIVA

Diferente da medição ativa, a medição passiva não é intrusiva, de modo que nenhum novo tráfego é introduzido ou gerado na rede [14]. Portanto, não há impacto no desempenho da rede. Muitas vezes as medições são realizadas pelos sistemas finais em comunicação, em que os protocolos envolvidos utilizam as próprias mensagens das aplicações como sondas. A medição passiva é comumente utilizada aplicações de reconhecimento de rede, como em [82]. Dispositivos específicos de coleta e análise estatísticas de tráfego realizam medições passivas para modelam o tráfego da rede, obtendo os parâmetros de estados e de desempenho [42]. A captura dos pacotes transmitidos por nó conectado ao elance é marcada com seu respectivo tempo, de modo que a precisão do relógio da entidade medidora é fundamental para se ter acurácia, muitas vezes sendo necessário hardware de relógios especializados para garantir maior precisão de tempo [89]. Nesta seção são discutidos os principais métodos e ferramentas de medição passiva.

2.3.1 *Métodos de Medição Passiva*

A medição passiva requer monitoramento do tráfego, de modo que os pontos de monitoramento passivo devem ser anexados ao elance, possibilitando a coleta de uma ampla amostra do tráfego observado. Nesses pontos, é possível coletar pacotes nas duas direções, em transmissões fim-a-fim entre servidores e clientes, onde as variações de roteamento têm um impacto mínimo [43]. Dessa forma, é possível observar

uma quantidade grande de tráfego e eventos ocorridos na rede. Nesse contexto, a escolha adequado do ponto de obervação na rede é um dos primeiros desafios da medição passiva.

A medição passiva por monitoramento pode ser realizada de duas formas: monitoramento de ponto único ou de vários pontos. O ponto único é mais adequado para realizar medições em redes locais (LAN), onde geralmente observa-se o tráfego no ponto em que liga a LAN à rede externa (WAN). Porém, existem limitações para o monitoramento de ponto único, pois não há possibilidade de lidar com medições de tempo. RTT e OWD somente podem ser obtidos via soluções de comunicação bidirecional, portanto, ao menos dois pontos devem existir. Geralmente, com medições de ponto único é possível realizar a contagem de diferentes eventos, como taxas precisas de bits ou pacotes, a quantidade de tráfego, disponibilidade da conexão e a distribuição da quantidade de diferentes tipos de protocolos que trafegam pelo ponto monitorado. Já o monitoramento de vários pontos possibilita expandir a quantidade de métricas que podem ser obtidas dos dados capturados. Por exemplo, a medição de atrasos bidirecional e unidirecional, instabilidades, taxas de perda de pacotes, utilização de elance, disponibilidade, monitoramento do tráfego e mudanças de rotas. Quanto maior o número de pontos observados na rede, maior a precisão. No entanto, medição por vários pontos é mais custosa, demandando maior complexidade e tempo para processamento dos dados capturados.

As métricas relacionadas ao tempo precisam de um relógio preciso em todos os pontos de monitoramento para obter medições mais confiáveis. A sincronização de relógios pode ser realizada com suporte de soluções de redes, e.g., protocolo NTP (*Network Time Protocol*) [88], controle de acesso CDMA (*Code Division Multiple Access*). Diferentemente, Gröhn [39] discute peculiaridades de se realizar essa sincronização através de receptor GPS (*Global Positioning System*) em um *cluster*.

Um outro desafio é a quantidade de dados coletados [90]. Os equipamentos necessários para coletar e análise dos dados de tráfego pode ser custosos, uma vez que necessitam de recursos de processamento e armazenamento para tratar os dados coletados [14]. Equipamentos de alto desempenho, com foco particular na velocidade da memória principal, muitas vezes são necessários [41]. Nesse contexto, é importante que haja comprometimento entre a quantidade de pontos de medição na rede e a quantidade de dados coletados, em simultâneo, em que se mantém a precisão das métricas [14]. A quantidade de dados coletados e armazenados pode ser reduzida de várias maneiras [90]. No momento da coleta dos dados, os dados não uteis para medição podem ser imediatamente removidos. Por exemplo, o descarte da carga útil dos pacotes, salvando apenas o cabeçalho dos mesmos, possibilita a redução significativa de dados, além de evitar questões relacionadas à privacidade dos dados coletados passivamente do tráfego de rede. Além disso, métodos de compactação ordinários, como `gzip`, também podem ser aplicados para reduzir informações redundantes dos dados coletados [90]. Outra forma é coletar apenas amostras do tráfego para realizar medições, não sendo necessário que todos os pacotes sejam capturados. Apenas uma parte do tráfego é suficiente para obter uma medida do estado atual da rede [14].

O tráfego a ser coletado pode ser selecionado por filtragem, classificação e amostragem [14]. Os filtros nos pacotes são aplicados com base em propriedades específicas dos mesmos [14]. Por exemplo, seleção de pacotes de um determinado protocolo, por valores de portas e endereços de origem/destino. Na classificação, os pacotes podem

ser identificados por classes de acordo com uma determinada propriedade de interesse [14]. Nesse caso, somente estatísticas das classes são calculadas e armazenadas. Na amostragem, os pacotes são coletados por critérios estatísticos [80]. Nem todos os pacotes são coletados, sendo feita uma seleção de aleatoriedade [14].

Existem várias técnicas para amostragem de tráfego. Geralmente, elas são divididas em duas categorias [80]: técnicas típicas de amostragem e técnicas específicas baseadas em algoritmos. As técnicas típicas de amostragem contêm amostragem sistemática, aleatória, estratificada e probabilística [41, 14]. A amostragem sistemática seleciona pacotes ou objetos do fluxo de dados a cada N intervalos [41]. Nesse caso, N pode ser um intervalo de tempo quando a amostragem for baseada no tempo [14], ou uma quantidade de pacotes. A amostragem aleatória seleciona cada pacote ou objeto de dados independentemente um do outro e com certas regras aleatórias [41]. Se a regra é simplesmente a probabilidade de $\frac{1}{N}$, a técnica é chamada de amostragem probabilística [14]. A amostragem estratificada divide o fluxo de pacotes em vários grupos ou tipos e, em seguida, seleciona uma única amostra de cada grupo com a mesma probabilidade [41]. As técnicas baseadas em algoritmos, embora mais complexas, são especializadas também em amostragem de tráfego. A escolha do melhor ou até um bom método de amostragem não é uma tarefa simples. O problema recorrente na amostragem é a coleta de uma quantidade de dados que seja mínima, mas representativa do comportamento do tráfego [50].

Segundo Lv *et al.* [80], não há um método de amostragem estatística melhor e mais útil, uma vez que sua escolha depende da aplicação, da quantidade e tipo de dados coletados. Uma desvantagem dos métodos de amostragens é que a análise de tráfego se torna mais complexa ou até mesmo impossível em alguns casos [90]. Se os pacotes capturados são de baixa representatividade, a medição passiva pode medir e analisar apenas uma parte do tráfego da rede. Portanto, a precisão e abrangência das medições dependem da qualidade das amostras de pacotes coletados e das escolhas nas quais os métodos estatísticos e analíticos são aplicados [42].

2.3.2 *Ferramentas de Medição Passiva*

As principais ferramentas de medição passiva são descritas na Tabela 3. A Figura 2 apresenta o digrama de fluxo *sankey* com as técnicas, as ferramentas, as métricas e protocolos utilizadas. As técnicas são se captura de pacotes (CP), análise de pacotes (AP) e requisição-resposta (RR). Maiores detalhes dessas ferramentas são apresentados a seguir.

As ferramentas amplamente utilizadas para medição passiva é **Wireshark** e o **Tcpdump**. O Tcpdump [59] foi desenvolvido no início dos anos 90 no Laboratório Nacional Lawrence Berkeley como uma ferramenta Unix (no caso WinDump para Windows) usada para coletar dados e pacotes de uma rede. Tcpdump exibe uma linha para cada quadro transmitido ou recebido. Em cada linha, o Tcpdump exibe a data e hora e informações derivadas dos cabeçalhos de protocolo contidos em um quadro. Se o quadro for um datagrama IP com uma carga útil que encapsule datagrama UDP ou segmento TCP, o Tcpdump exibirá informações sobre a origem e o destino do quadro, considerando endereços IP e portas de origem e destino. A ferramenta não faz análise dos dados capturados, porém fornece muitas opções em que os detalhes dos pacotes capturados podem ser visualizados em vários formatos. Por exemplo, a

Tabela 3: Principais ferramentas de medição passiva.

Ref.	Ferramenta	Descrição
[59]	Tcpdump	Ferramenta UNIX usada para coletar dados de uma rede.
[75]	Wireshark	Semelhante ao Tcpdump, porém possui uma interface gráfica e realiza análise no tráfego coletado.
[17]	NetFlow	Implementado pela primeira vez em roteadores Cisco, é um software de medição, análise e coleta de tráfego. Usa amostragem sistemática para amostrar o tráfego.
[120]	SNMP	Protocolo de gerenciamento de rede mais utilizado. O SNMP organiza as métricas medidas em uma estrutura MIB.
[117]	SPAND	Usa informações coletadas passivamente de vários nós em uma rede para medir as condições da rede.

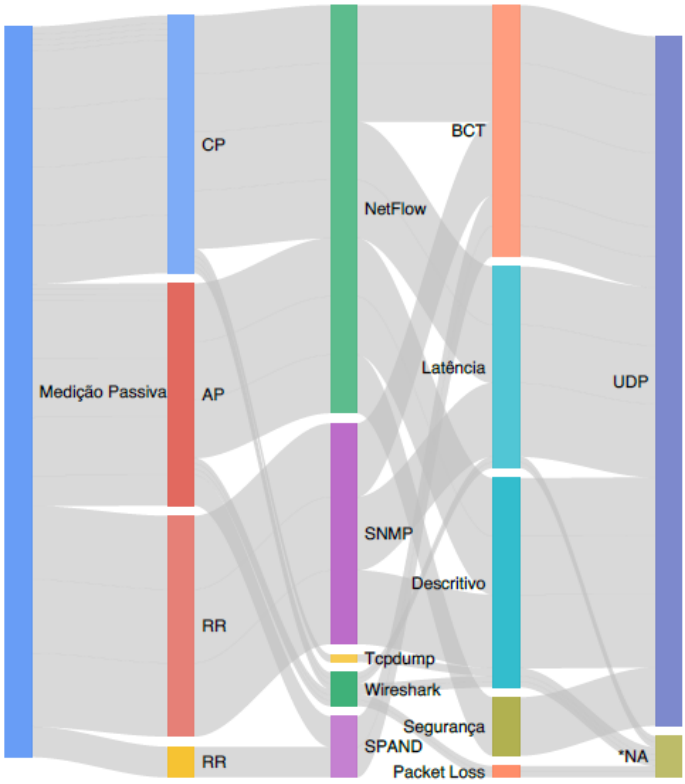


Figura 2: Diagrama de fluxo *sankey* as técnicas, as principais ferramentas, as métricas e os protocolos da pilha utilizados em medições passivas.
Fonte: do autor

opção `-t` exibe a marcação de tempo em formato legível por humanos, `-X` exibe em formatos Hex e ASCII o conteúdo do pacote capturado, `-v`, `-vv` e `-vvv` aumentam as verbosidade ao expor dados do pacote. Uma vez coletados dados dos pacotes pelo Tcpdump, pode-se então realizar análises e extrair métricas da rede.

Gerald Combes desenvolveu **Ethereal** em 1997, atualmente denominado **Wireshark**. Essa ferramenta é gratuita, disponível para Windows e Unix, e é particularmente amigável ao usuário. Diferente de Tcpdump, ela possui uma interface gráfica de usuário para auxiliar na navegação dos pacotes capturados e na análise do tráfego. A ferramenta é capaz de ler a saída binária do Tcpdump, bem como usar os mesmos filtros do Tcpdump para capturar e exibir seletivamente os pacotes. O Wireshark oferece muitas opções de decodificação dos dados dos protocolos, mais de quatrocentos no total [75]. Dentre as ferramentas de captura de tráfego de redes gratuitamente disponíveis, Wireshark é uma das melhores. Sendo criado com base nas propriedades do Tcpdump (filtragem, etc.), além de facilitar a compreensão dos fluxos de pacotes em andamento através de sua visualização, o Wiresharak facilita a solução de problemas da rede ao prover análise de desempenho e comportamento do tráfego. Um comparativo entre Tcpdump e Wireshark pode ser visto em [38, 37].

A ferramenta **NetFlow** [17] foi implementado inicialmente em roteadores Cisco, sendo um software de medição de tráfego [50, 19, 80, 135]. A ferramenta utiliza um método de medição passiva para coletar, amostrar e analisar o tráfego de rede. O NetFlow usa amostragem sistemática para amostrar o tráfego de rede [135], onde os roteadores mantêm um cache que contém registros de fluxos que descrevem o tráfego encaminhado. Esses registros são então exportados via UDP para um computador que os analisa e arquiva. As análises podem ser realizadas, por exemplo, para detectar problemas de segurança, perfilar tráfego ou construir matriz de tráfego origem-destino para tomografia em rede [135, 19]. Por outro lado, o método sistemático de amostragem é propenso a desvios de amostra e problemas de desempenho [50, 80]. Pacotes transferidos periodicamente podem ser sistematicamente deixados de fora pela seleção. Dessa forma, a amostragem aleatória é geralmente preferida à amostragem sistemática [50].

SNMP (*Simple Network Management Protocol*) [120] é um protocolo de gerenciamento de rede amplamente utilizado. O protocolo organiza as métricas medidas em uma estrutura hierárquica de dados denominada MIB (*Management Information Base*). Os roteadores que oferecem suporte ao SNMP implementam MIBs públicas e específicas do fornecedor. Medições são realizadas através das MIBs associadas as interfaces de enlace, por exemplo, pacotes e bytes recebidos e transmitidos ou descartados e erros de transmissão.

A ferramenta **SPAND** (*Shared Passive Network Performance Discovery*) [117] usa dados coletados passivamente de vários nós para medir as condições da rede. Os dados de desempenho são coletados por nós clientes e nos de captura na rede. Os relatórios de desempenho enviados pelos clientes são baseados em observações das aplicações e podem não ter detalhes necessários para fornecer estimativas precisas da largura de banda disponível. Os nós de captura de pacotes, que usam *BSD Packet Filter* (BPF) McCanne e Jacobson [86] para observar o tráfego de rede de um grupo de nós, são os principais meios de coleta de dados. Esses nós não estão nos pontos finais do caminho e, portanto, precisam aplicar heurísticas para inferir propriedades fim-a-fim, como largura de banda alcançável. O nó de captura de pacotes é responsável pelo

processamento dos dados antes de enviar o relatório de desempenho ao servidor. SPAND mantém um repositório central de medições em um servidor de desempenho, que pode ser consultado por qualquer aplicação na rede. No SPAND, há ênfase na obtenção de métricas relacionadas ao desempenho que uma aplicação pode obter, em vez da disponibilidade total dos recursos de rede. Uma das métricas de interesse fornecida é a de largura de banda alcançável, semelhante à métrica de BTC (*bulk transfer capacity*) [85].

2.3.3 *Medição Passiva em Protocolos de Transporte*

Os protocolos TCP [101] e QUIC [55], para prover transmissões confiáveis de dados, necessitam detectar eventos errôneos nas transmissões para corrigi-los. Para isso, contam com mecanismos passivos não intrusivos que realizam medições sobre transmissão para então inferir o estado da rede. Comumente, ocorrem a medição de latência e detecção de pacotes perdidos para mitigar os efeitos do congestionamento no caminho, ao passo que possibilita a retransmissão dos pacotes perdidos.

Na medição de latência no protocolo TCP, o emissor mede o tempo bidirecional, considerando o tempo que o pacote leva para percorrer o caminho da ida entre a origem (emissor) e o destino (receptor), bem como o tempo do caminho da volta, sinalizando que o pacote foi recebido com sucesso pelo receptor. Para realizar o cálculo, o emissor embute no campo de opções do pacote a marcação de seu tempo de envio. O receptor, ao receber o pacote, copia essa marcação de tempo e embute de volta nas opções do pacote de reconhecimento (ACK). Assim, o emissor ao receber o ACK, obtém a maração atual e faz a diferença com o tempo de envio. Este tempo decorrido é, portanto, o tempo de RTT (*round-trip time*). Essa medição de latência é um recurso importante que é utilizado em várias funções do protocolo TCP, como determinar os temporizadores de retransmissão de pacotes perdidos. O temporizador de retransmissão RTO (*retransmission timeout*) é uma estimativa de tempo limite que emissor deverá esperar para receber os pacotes de confirmação ACK. O RTO é determinado pelo emissor da seguinte forma:

$$\mathrm{RTO} = \mathrm{sRTT} + \max(\mathrm{G}, 4 * \mathrm{RTTVAR}), \tag{1}$$

onde sRTT (*smoothed round-trip time*) é uma média móvel, RTTVAR (*round-trip time variation*) é um desvio, e G granularidade em segundos de um relógio [96].

A média móvel e o desvio são calculados da seguinte forma:

$$\mathrm{RTTVAR} = (1 - \beta) \times \mathrm{RTTVAR} + \beta \times |\mathrm{sRTT} - \mathrm{RTT}|, \tag{2}$$

$$\mathrm{sRTT} = (\alpha \times \mathrm{sRTT}) + ((1 - \alpha) \times \mathrm{RTT}), \tag{3}$$

com pesos $\alpha = 1/8$ e $\beta = 1/4$, como sugerido por Jacobson [57].

Os métodos de medição de latência e perdas de pacotes no QUIC [3, 54] são similares aos do TCP. Apenas algumas diferenças algorítmicas, pois a estrutura do pacote QUIC difere do TCP. Por exemplo, o QUIC divide os dados em quadros independentes. Assim, cada pacote pode carregar diferentes tipos de quadros, como quadros de dados, controle, reconhecimento, etc. Cada pacote QUIC carrega um novo número de sequência, incluindo aqueles que transportam dados retransmitidos. Isso possibi-

lita o emissor distinguir o ACK de uma transmissão original de um ACK de uma retransmissão, evitando assim o problema de ambiguidade de retransmissão do TCP [65]. Ao reconhecer o pacote, o QUIC envia quadros ACK com o número do maior pacote reconhecido, bem como uma lacuna com os números de sequência precedente e ainda não reconhecidos. Neste cenário, as amostras de RTT é gerada usando apenas o maior pacote reconhecido no quadro ACK recebido.

A retransmissão por tempo requer que o emissor estime o tempo limite para uma retransmissão, denominado PTO, de forma similar ao RTO do TCP. A diferença é que no QUIC inclui-se o atraso máximo de confirmação esperado, em vez de usar um tempo limite mínimo fixo. O cálculo do PTO (*probe timeout*) é dado da seguinte forma:

$$\mathrm{PTO} = \mathrm{sRTT} + \max(4 * \mathrm{RTTVAR}, \mathrm{kGranularity}) + \mathrm{maxAckDelay}, \quad (4)$$

onde minRTT é mínimo RTT observado, maxAckDelay é o tempo máximo que o receptor por esperar para emitir uma confirmação, e kGranularity evita que o temporizador expire imediatamente. a média móvel sRTT e RTTVAR são determinadas conforme o Algoritmo 1. A variável de ackDelay é o menor tempo entre o maxAckDelay e o Ack Delay in ACK Frame, que é o tempo efetivamente esperado.

Algorithm 1 Compute sRTT e o RTTVAR [54]

```
ackDelay = min(Ack Delay in ACK Frame, maxAckDelay)
adjustedRTT = latestRTT
if (minRTT + ackDelay) < latestRTT then
    adjustedRTT = latestRTT − ackDelay
end if
sRTT = 7/8 × sRTT + 1/8 × adjustedRTT
RTTVARSample = abs(sRTT − adjustedRTT)
RTTVAR = 3/4 × RTTVAR + 1/4 × RTTVARSample
```

2.4 CONSIDERAÇÕES FINAIS

Medições de rede são métodos essenciais para lidar com tráfego de dados, seja para fins de gerenciamento e operação de rede, seja para tomada de decisões de protocolos na pilha dos sistemas finais. Nesse contexto, este capítulo possibilitou uma visão das principais técnicas e ferramentas para medições ativa e passiva, por salto ou fim-a-fim, considerando, principalmente, métricas de largura de banda, latência e monitoramento. A diferença fundamental entre a medição passiva e ativa, como mencionado, é que a passiva é não-invasiva. A medição ativa, ao introduzir pacotes de sonda na rede, pode gerar perturbação no tráfego corrente, gerando uma pressão reversar que pode afetar as próprias medições. Por outro lado, a medição passiva é mais complexa, visto que a sua precisão é dependente de métodos estatísticos e analíticos, além da qualidade das amostras.

Em geral, os protocolos de transporte usam medições passivas. Sua complexidade de medição se dá ausência de dados assertivos e precisos quanto ao estado de gargalo. As medições bidirecionais, como RTT, carregam ruído ao considerar a latência do caminho da volta. No caso dos mecanismos de SBD, foco de investigação deste trabalho, as medições passivas são realizadas pelos próprios protocolos de transporte, contudo, de forma unidirecional. Essas medições são fundamentais para a determina-

ção de métricas estatísticas sobre os subfluxos e, com isso, possibilitando a detecção dos compartilhamento de gargalos.

3

CONEXÕES RÁPIDAS COM UDP

Neste capítulo são apresentados os protocolos de conexões rápidas na Internet baseados em UDP, os quais têm o objetivo de reduzir a latência de comunicação fim-a-fim. A seção seguinte apresenta o protocolo QUIC (*Quick UDP Internet Connection*) que tem como motivação melhorar o desempenho de transmissões baseadas em HTTP(S). Na sequência, a Seção 3.2 apresenta o protocolo MPQUIC que é uma extensão do QUIC para possibilitar transmissões *multipath*, possibilitando tanto o aumentar a vazão e diminuir a latência fim-a-fim. Por fim, a Seção 4.4 traz as considerações finais deste capítulo.

3.1 QUIC - *quick udp internet connection*

O protocolo QUIC foi proposto e implementado pela *Google* e hoje é executado em milhares de servidores HTTP para servir clientes, originalmente os navegadores *Chrome*, em tráfego multimídia da aplicação *Youtube*. O QUIC é um protocolo de transporte de dados projetado para atender vários objetivos [110], como facilidade de capacidade de implantação, segurança, redução de atrasos no estabelecimento de conexões, tratamento de bloqueios HoL (*Head-of-Line*). O protocolo fornece baixa latência, suporte a transmissão criptografada, multiplexação de fluxos e transferência de dados em ordenada e confiável. Sua motivação inicial foi prover melhor desempenho que o protocolo TCP em transmissões de tráfego HTTPS [76]. A abordagem em camadas do QUIC é mostrada na Figura 3. Importante destacar que ele é um protocolo implementado na Camada de Aplicação, incorporando recursos de controle de congestionamento e recuperação de perdas semelhantes aos mecanismos previstos no protocolo TCP na Camada de Transporte, além de fornecer recursos avançados de sinalização entre cliente e servidor. O protocolo QUIC combina *handshake* criptográfico e de transporte para configurar uma conexão segura ao mesmo tempo em que diminui a latência do estabelecimento da conexão.

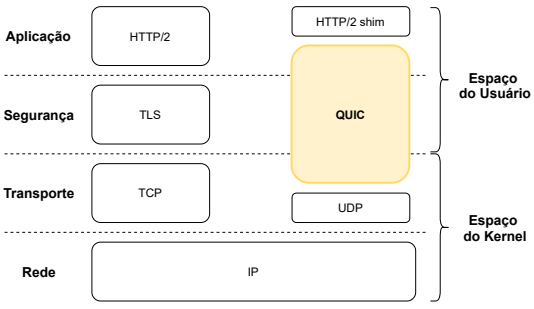


Figura 3: QUIC em uma pilha HTTPS.
Fonte: Imagem adaptada de [76].

3.1.1 *Estabelecimento de Conexões*

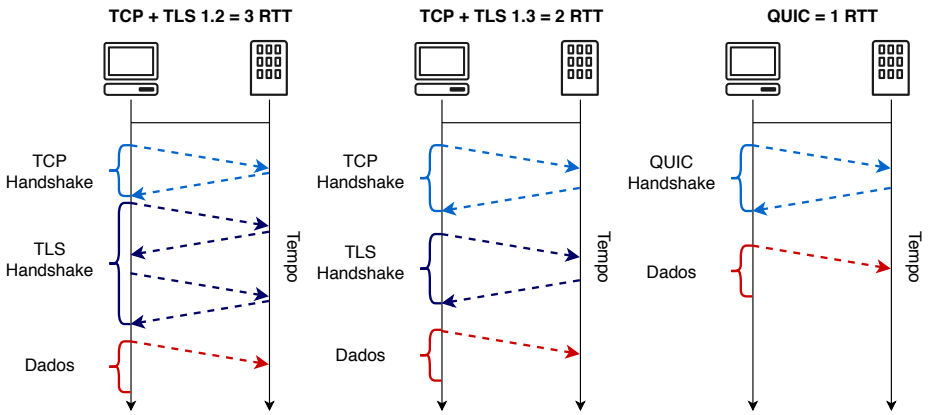
No protocolo TCP, o estabelecimento de conexão é realizado através de uma negociação em 3 vias entre cliente e servidor, cujo processo é conhecido como *handshake*. O *handshake* de três vias (ou 3 etapas) é realizado da seguinte forma: (1) Sempre iniciada pelo cliente, enviando um pedido de uma conexão com o servidor através de um segmento TCP de controle (sem carga útil da aplicação) com a *flag* SYN ativa, juntamente com um número de sequência aleatório; (2) Em resposta, o servidor responde com um pacote de controle com as *flags* SYN e ACK ativas, com seu número de sequência aleatório, com o número de reconhecimento baseado no número de sequência recebido; (3) para finalizar, o cliente responde com um pacote ACK com o número de reconhecimento baseado no número de sequência recebido do pacote SYN/ACK do servidor. Essa apresentação entre cliente e servidor está presente em todas as conexões TCP. Geralmente, o estabelecimento da conexão leva 1 RTT para ser concluído, uma vez que na maioria das implementações do TCP, o último pacote (ACK) já carrega dados da aplicação.

Com aumento do tráfego na Internet, por consequência o maior uso e compartilhamento de recursos de rede, há uma demanda em direção à redução de latência transmissão e à segurança e privacidade dos dados. Para se obter uma comunicação segura fim-a-fim, faz-se necessário o uso de mecanismos de confidencialidade, integridade, autenticidade e não-repúdio. Na arquitetura do protocolo TCP não foi prevista a comunicação segura, portanto, protocolos de segurança habilitam comunicação segura em uma conexão TCP. Nesse caso, SSL (*Secure Sockets Layer*) [36] e TLS (*Transport Layer Security*) [22, 107, 106] são as principais referências. O SSL foi um protocolo bem conhecido por fornecer originalmente segurança à comunicação fim-a-fim. TLS é o sucessor do SSL, fornecendo comunicação segura para uma variedade de aplicações, como navegadores web, VPNs, entre outros. A conexão TLS pode ser dividida em duas fases: um *handshake* inicial e a transferência de dados da aplicação. Na fase de *handshake*, o cliente envia uma mensagem *Client Hello*, identificando a versão TLS usada e os *CipherSuites* suportados pelo cliente, que é um conjunto de um ou mais soluções de criptográficas. O servidor TLS responde com uma mensagem *Server Hello*, a qual contém o *CipherSuite* escolhido pelo servidor a partir da lista fornecida pelo cliente, juntamente com o identificador da sessão e seu certificado digital. As mensagens seguintes são usadas para autenticação por pares e cálculo das chaves privadas. Dessa forma, logo na terceira via (ACK) do estabelecimento de conexão TCP, inicia-se o *handshake* TLS para troca de chaves criptográficas e certificados digitais. Após o *handshake*, a segunda fase da conexão prevê a transmissão segura de dados entre os nós, utilizando o material criptográfico negociado.

No protocolo QUIC, o estabelecimento de conexão é realizado de forma diferente do TCP. Inicialmente, o cliente não tem informação sobre o servidor, portanto, envia-se uma mensagem CHLO (*inchoate client hello*). Na sequência, o servidor responde com uma mensagem REJ (*reject*), contendo uma configuração que inclui o valor público de sua chave, do tipo *long-term Diffie-Hellman*, uma cadeia de certificados, uma assinatura do servidor, um *token* de endereço de origem e uma marcação de data/hora do servidor [76]. Ao receber o pacote REJ, o cliente responde com uma mensagem CHLO completa, contendo suas informações de criptografias do valor público de sua chave, do tipo *Ephemeral Diffie-Hellman*. Em seguida, se o *handshake* for bem-

sucedido, o servidor responderá com uma mensagem SHLO (*server hello*), contendo a informação de criptografia do valor público, agora do tipo *Ephemeral Diffie-Hellman*. E com base nas informações de criptografia, o cliente e servidor podem calcular as chaves finais para continuar a transmissão de forma segura. Uma vez armazenado esse material criptográfico nos nós, os estabelecimentos de conexões posteriores entre esses nós serão de latência o-RTT. Nesse caso, o cliente contém as informações de chaves iniciais da conexão, de modo que o mesmo pode enviar dados criptografados desde o primeiro pacote, sem que o servidor o responda [76].

A Figura 4 considera diferentes abordagens para o estabelecimento de conexões seguras entre dois pares, cliente e servidor, quando não é realizada nenhuma comunicação segura prévia. Portanto, ambos nós não possuem as chaves e certificados um do outro. Na versão TLS amplamente utiliza, TLSv1.2, levam-se 2 RTTs para que esse *handshake* seja concluído, totalizando 3 RTTs até que seja possível iniciar a troca de mensagens entre as aplicações cliente e servidor, como mostra a Figura 4(a). A versão TLSv1.3 reduz o procedimento de troca de chaves para 1 RTT, totalizando 2 RTTs para ser concluído o estabelecimento da conexão TCP sob TLS, como mostra a Figura 4(b). Já o protocolo QUIC, atendendo as demandas tanto de transmissão segura e quanto de redução de latência de comunicação, realiza o estabelecimento de conexão e a troca de certificados e chaves levando apenas 1 RTT, conforme mostra na Figura 4(c).

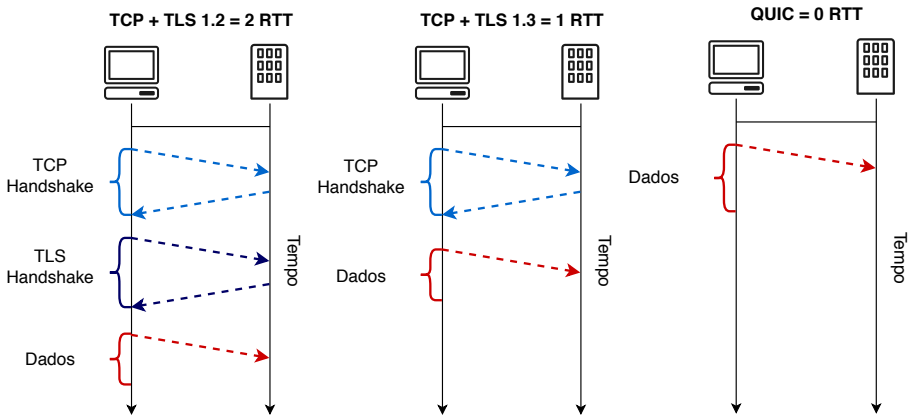


(a) TLS v1.2 (b) TLS v1.3 (c) QUIC

Figura 4: Estabelecimento conexão segura sem ter ocorrido comunicação prévia entre cliente e servidor.
Fonte: Imagem adaptada de [103].

Após estabelecer a conexão pela primeira vez, os protocolos nos pares armazenaram em cache as chaves e certificados criptográficos, não sendo mais necessário a troca dos mesmos entre cliente e servidor. Assim, os protocolos reduzem o tempo para iniciar a transmissão, reutilizando as chaves em cache. Dessa forma, os protocolos TLSv1.2 conseguem reduz em 1 RTT o tempo para estabelecer uma comunicação com o cliente pela segunda vez, como mostra a Figura 5(a). Na versão TLSv1.3 necessita apenas do *handshake* do TCP para iniciar a transmissão de dados, como mostra a Figura 5(b). No QUIC, o cliente pode estabelecer uma conexão segura sem qualquer latência adicional, ao contrário do que ocorre no TCP, enviando dados das aplicações imediatamente já na primeira via, como mostra na Figura 5(c), possibilitando redução de latência significativa em uma transmissão. Caso o material criptográfico

reutilizado pelos pares tenha sido expirado, restabelece-se uma nova conexão segura como descrito no parágrafo anterior.



(a) TLS v1.2 (b) TLS v1.3 (c) QUIC

Figura 5: Estabelecimento de uma nova conexão segura após já ter ocorrido comunicação prévia entre cliente e servidor.
Fonte: Imagem adaptada de [103].

3.1.2 *Pacotes QUIC*

Um pacote QUIC é encapsulado em um datagrama UDP, como ilustra a Figura 6. A estrutura do pacote prevê um cabeçalho público (não-criptografado) com dados de controle para os sistemas finais clientes e servidor envolvidos na conexão, seguido pela carga útil (*payload*) que é criptografada.

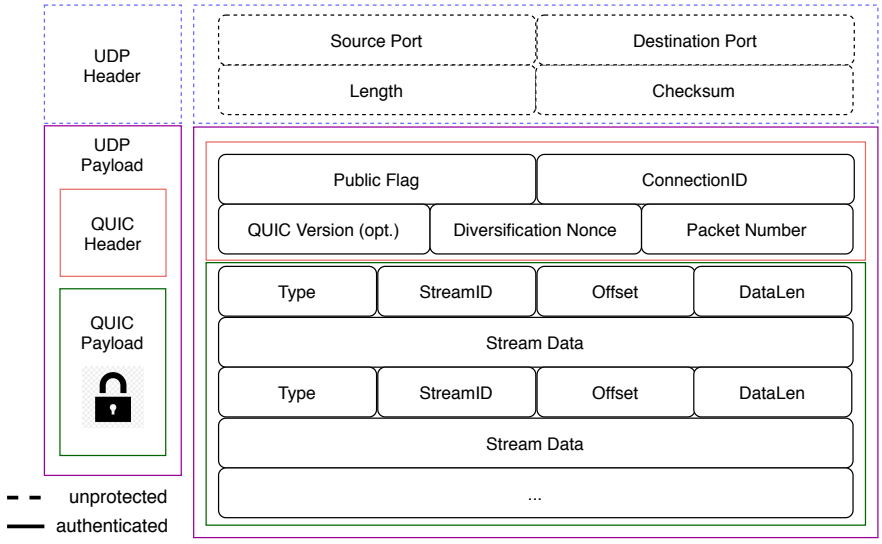


Figura 6: Estrutura de um pacote QUIC. Em vermelho destaca-se o cabeçalho público autenticado, mas não criptografado. Em verde indica-se a carga útil que é criptografada. Essa estrutura de pacotes vem sendo evoluída à medida em que o protocolo QUIC passa por padronizações junto ao IETF.
Fonte: imagem adaptada de [125, 55].

O cabeçalho possui campos descriptografados, sendo úteis para fins de eventuais filtragens em *middle-boxes*, suporte à roteamento de fluxos em equipamentos de rede, entre outros. Os campos do cabeçalho são compostos por *Flags*, ID da conexão, Versão, Nonce, e Número do Pacote. Os códigos do campo Flags indica a presença do campo ID da conexão e o comprimento do campo Número do Pacote. O campo de versão somente está presente nos pacotes de *handshake*, sendo usada pelo cliente para propor uma versão do QUIC ao servidor durante o estabelecimento de uma conexão. O número de pacote é usado para identificar o pacote no fluxo de transmissão, com valor único, inclusive em pacotes retransmitidos. Isso elimina a necessidade de um mecanismo separado para distinguir confirmações de retransmissões das transmissões originais, evitando o problema de ambiguidade de retransmissão observada no TCP [76]. O campo *Diversification Nonce* é um número utilizado uma única vez cuja função é adicionar entropia à geração de chaves [55], sendo usado apenas no estabelecimento da conexão, não estando mais presente no pacote depois disso. Para autenticar e descriptografar pacotes, os nós nas extremidades da conexão usam o número de pacote como um *nonce* do pacote. Dessa forma, o número de pacote é público, de modo a suportar a descriptografia de pacotes recebidos fora de ordem, semelhante ao que ocorre em DTLS [108]. Os pacotes de *Frame* são responsáveis por transportam os dados reais do aplicativo dentro da conexão. A carga útil está localizada diretamente após o cabeçalho e formatada como mostra a Figura 6.

Na carga útil do pacote, a parte criptografada, o campo `Type` especifica o tipo de quadro (`Frame`), relacionado ao fluxo de dados (`Stream Data`) encapsulado. Dentro de um mesmo pacote QUIC podem haver diferentes `Frames`, sendo os mais comuns os de `Stream Frame`, que carregam dados das aplicações, e os de `Acknowledge Frames` usados para sinalizar que um pacote foi recebido ou está ausente. O tipo `Congestion Feedback Frame` foi proposto para fornecer retroalimentação entre as partes quanto ao congestionamento observado no caminho. Contudo, é um campo que ainda não tem sido usados por implementações do protocolo. De forma similar ao protocolo SCTP (`Stream Control Transmission Protocol`), o QUIC pode multiplexar os fluxos de dados das aplicações, por exemplo, diferentes requisições HTTP, dentro de uma única conexão. Cada fluxo multiplexado possui um identificador (`Stream ID`), um `offset` de deslocamento dos bytes transmitidos, na definição de seu comprimento (`DataLen`). Como cada fluxo tem seu próprio identificador e o número de sequência no cabeçalho através do campo de `Offset`, a dependência entre fluxos de uma conexão é reduzida. Nesse caso, se um pacote for perdido, apenas afetaria o fluxo de sequência deste pacote e não toda a conexão, mitigando o problema de bloqueio de HoL entre os fluxos.

Assim como TCP, o protocolo QUIC possui mecanismos internos para controle de congestionamento e de fluxo. No entanto, a implementação desses mecanismos e do próprio protocolo QUIC não exige alterações no núcleo dos sistemas operacionais. Pelo fato de ser um protocolo implementado como componente de software no espaço de usuário, sua adoção, implantação e alteração são mais simples para as aplicações.

3.2 MPQUIC - *multipath quic*

MPQUIC é um protocolo que estende o QUIC, possibilitando que fluxos multiplexados internamente sobre uma conexão possam ser remutiplexados sobre subfluxos, os quais são conexões específicas que correm sobre diferentes caminhos fim-a-fim. Embora seja um protocolo de transmissão por múltiplos caminhos, seu esquema de multiplexação é um pouco mais sofisticado que o do protocolo MPTCP (*Multipath TCP*) [35].

3.2.1 *Caminhos no MPQUIC*

Como ilustra da Figura 7, enquanto o QUIC multiplexa a transmissão em fluxos de dados das aplicações dentro de uma única transmissão UDP, o MPTCP recebe um único fluxo de dados das aplicações e os dividindo em pacotes que são multiplexados sobre os diferentes subfluxos TCP por um mecanismo de escalonamento. O protocolo MPQUIC, por sua vez, combina as duas abordagens, onde o escalonador multiplexa os vários fluxos de dados vem da aplicação em várias subfluxos UDP.

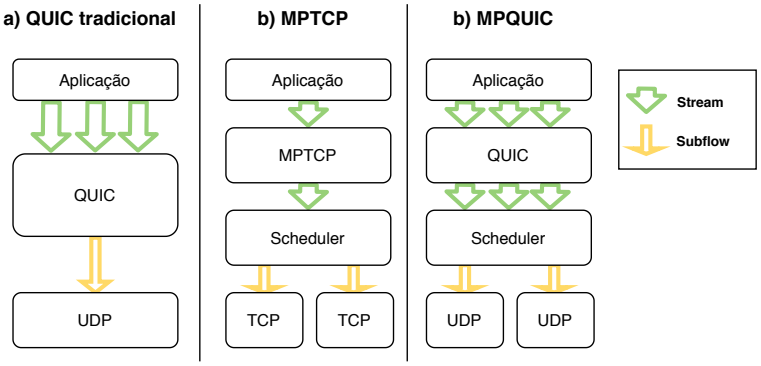


Figura 7: Comparação dos fluxos QUIC, MPTCP e MPQUIC.
Fonte: imagem adaptada de Viernickel *et al.* [125].

Ao possibilitar transmissão por múltiplos caminhos, o MPQUIC pode usufruir de benefícios que o QUIC provê na transmissão fim-a-fim. Uma das vantagens é a multiplexação por completo de solicitações e respostas, que possibilita a divisão de mensagens em quadros independentes para transmissão. Isso permite o receptor receba os quadros por caminhos diferentes, podendo assim, tomar melhores decisões e reduzindo os efeitos causados pela perda de pacote e latência, como pode ser observado na Figura 8. Para isso ser possível, cada caminho possui seu próprio identificador que, quando combinado ao número do pacote no cabeçalho público, permite que o MPQUIC reconheça pacotes que foram recebidos de caminhos diferentes. A divisão por pacote traz outra vantagem que é a diminuição dos problemas causados por bloqueios HoL. Se um pacote for perdido, apenas afetará o fluxo que está transmitindo o pacote e não toda a conexão. No exemplo da figura, se o pacote 6 do caminho 1 for perdido, ele não afetará outros fluxos compostos pelos pacotes {1, 2, 3} e {4, 5}, tampouco os fluxos do caminho 2.

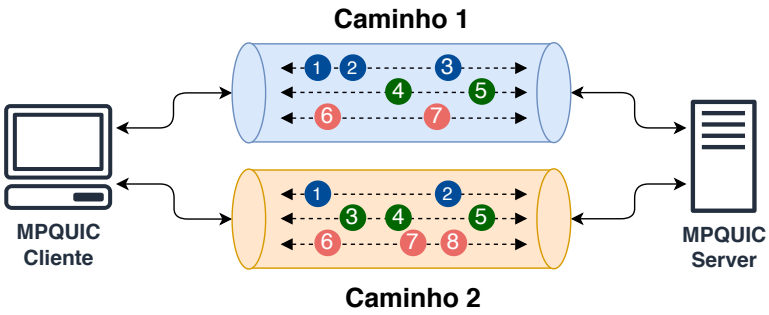


Figura 8: Exemplo de uma conexão MPQUIC sobre dois caminhos.
Fonte: imagem adaptada de [77].

3.2.2 *Mecanismos Internos*

Para que os pacotes sejam inseridos nos subfluxos, requer a atuação de quatro principais componentes: o controle de fluxo, o controle de congestionamento, detecção de perda, e o escalonador de pacote.

3.2.2.1 *Controle de Fluxo*

O controle de fluxo, assim como no TCP, limita o envio de pacotes para não sobrecarregar o receptor, considerando a disponibilidade do seu *buffer* de recepção. O protocolo QUIC usa dois mecanismos de controle de fluxo: (1) controle de fluxo do fluxo; (2) controle de fluxo do nível da conexão. No nível do fluxo, o receptor notifica o emissor da quantidade máxima que pode receber por fluxo de dados usando o quadro. Além desse controle no nível do fluxo, o QUIC implementa também o controle no nível da conexão, limitando dinamicamente a quantidade de dados que o emissor pode enviar, conforme a capacidade suportada pelo *buffer* do receptor. O mesmo conceito é adotado pelo MPQUIC com a adaptação da utilização de estatísticas básicas (e.g., *One-way delay* e quantidade e de pacotes enviados) para determinar o deslocamento das atualizações da janela de recepção e seu intervalo de envio [20, 21].

3.2.3 *Controle de Congestionamento Acoplado*

Para se obter um compartilhamento de recursos mais justos durante a transmissão de dados, os protocolos de transporte implementam o **controle de congestionamento**. Tanto TCP quanto QUIC aplicam o algoritmo CUBIC [40]. Entretanto, o CUBIC não obtém uma boa performance em protocolos de múltiplos caminhos [131]. Nesse caso, o controle de congestionamento do protocolo MPQUIC é adaptado do protocolo MPTCP, através do algoritmo OLIA (*Opportunistic Linked-Increases Algorithm*) [21]. O controle de congestionamento é conduzido por subfluxo, ou seja, cada subfluxo possui uma janela própria de congestionamento. No entanto, o conjunto de subfluxos é tratado de forma acoplada sob um controle global da conexão, com o intuito de prover justiça no uso de gargalos compartilhados com outras transmissões que não são de múltiplos caminhos [104].

De acordo o algoritmo OLIA [68], para cada ACK recebido no subfluxo r, a janela de congestionamento w_r é aumentada pela adição de dois termos:

$$\left(\frac{w_r/\tau_r^2}{\left[\sum_{i\in\mathcal{R}}\left(w_i/\tau_i\right)\right]^2}\right)+\left(\frac{\alpha_r}{w_r}\right). \tag{5}$$

O primeiro termo é um equilíbrio de congestionamento ideal, em que τ_r é o tempo de ida e volta observado em r. O segundo termo garante a capacidade de resposta para reagir às mudanças no w_r atual. O parâmetro α_r é específico para cada subfluxo r e aplicado para deslocar o tráfego entre os subfluxos. Portanto, o cálculo para aumento de uma janela de um subfluxo utiliza a informação da janela de todos os subfluxos disponível no momento.

3.2.4 *Retransmissão Rápida*

O terceiro componente é o de detecção de perda e retransmissão de pacotes perdidos. No MPQUIC, cada subfluxo implementa seu próprio esquema que é baseado em ACK e controle de congestionamento [54]. O MPQUIC usa o `Ack Frame` para confirmar que os dados foram recebidos, caso o receptor demore para confirmar ou não confirme o recebimento, ocorrerá estouro do temporizador RTO (*retransmission timeout*). Esse temporizador de limite, assim como no TCP, é baseado no RTT, contudo, de cada subfluxo. Após uma perda detectada, os quadros transportados pelo pacote perdido são agrupados em outro pacote com um novo número de sequência de pacotes. Portanto, os números de sequência de pacotes não se repetem em uma conexão [125].

3.2.4.1 *Escalonamento de Pacotes*

O quarto e último mecanismo é o escalonador de pacotes, sendo um dos componentes principais de qualquer arquitetura de agregação de banda [69]. Assim que vários caminhos estão ativos pelos subfluxos, o escalonador de pacotes seleciona qual é o melhor caminho para transmitir o pacote da vez, de modo que os pacotes sejam recebidos e entregues de forma confiável e ordenada. Portanto, uma decisão errada de escalonamento pode impactar no desempenho da transmissão fim-a-fim. Os pacotes disponíveis são inseridos em uma fila de envio e, conforme a política de escalonamento implementada, os pacotes são retirados dessa fila e inseridos nas respectivas filas dos caminhos disponíveis.

Atualmente, há duas políticas de escalonamento disponível no MPQUIC, o Lowest-RTT-First (Low-RTT) e o Round-Robin (RR). O escalonador padrão é o Low-RTT, inspirado no escalonador do MPTCP, se baseia no tempo de ida e volta observado em cada subfluxo e seleciona como melhor subfluxo aquele que possui o menor sRTT e esteja disponível, i.e., com espaço na janela de congestionamento para realizar a transmissão daquele pacote. O escalonador RR se baseia em uma seleção circular, de modo que os pacotes são inseridos alternadamente sobre os caminhos disponíveis, portanto, não há critério de seleção. Comparado a outras políticas, RR não é recomendada para fins de obtenção de desempenho, uma vez que não seleciona caminhos de melhores métricas de desempenho [78].

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os protocolos QUIC e MPQUIC, que são protocolos de conexões rápidas na internet baseados em UDP. Enquanto o QUIC tem por motivação reduzir a latência na comunicação em relação ao HTTP tradicional com múltiplos fluxos sobre uma conexão de caminho único, o protocolo MPQUIC estende QUIC habilitando a transmissão de múltiplos fluxos sobre uma conexão de múltiplos caminhos.

O protocolo MPQUIC é um protocolo recente de 2017, havendo poucas implementações disponíveis. Há oportunidades de inovação em seus componentes e mecanismos internos devido à sua recenticidade, mas, sobretudo, pelo fato de que muitas investigações em estado maduro no protocolo MPTCP ainda não foram exploradas em iguais esforços. Esse é o caso do mecanismo SBD (*Shared Bottleneck Detecction*), objeto de pesquisa neste trabalho, que foi originalmente proposto para detectar gargalos compartilhados entre subfluxos TCP pertencentes a uma conexão MPTCP. O MPQUIC, no melhor do nosso conhecimento, ainda não possui o suporte de um SBD em suas conexões de subfluxos QUIC/UDP. Portanto, os impactos de tal mecanismo, enquanto já são conhecidos sobre o protocolo MPTCP, não desconhecidos em transmissões MPQUIC. O capítulo seguinte apresente MPQUIC-SBD, como contribuição deste trabalho.

4

MPQUIC-SBD: DETECÇÃO DE GARGALOS COMPARTILHADOS NO MPQUIC

Neste capítulo é apresentado o desenvolvimento de um mecanismo SBD (*Shared Bottleneck Detection*) para detecção de gargalo compartilhado, conforme o padrão RFC 8382 [46], junto ao protocolo MPQUIC. A seção seguinte descreve as principais partes do mecanismo, conforme metodologia prevista no padrão, considerando as medições de latência nos subfluxos, métricas estatísticas e agrupamentos de subfluxos. Em seguida, a Seção 4.2 apresenta aspectos importantes da implementação do mecanismo junto à implementação do protocolo MPQUIC em Linguagem Go. Na Seção 4.3 são discutidos os principais trabalhos relacionados. Por fim, a Seção 4.4 traz as considerações finais deste capítulo.

4.1 MÉTODO SBD CONFORME RFC 8382

O método SDB considerado neste trabalho é descrito na RFC 8382 [46], o qual prevê três importantes etapas: (1) medições de latência de unidirecional (OWD) em cada subfluxo pertencente a uma conexão de múltiplos caminhos; (2) extração de métricas estatísticas a partir de amostras de OWD medidos; (3) agrupamento dos subfluxos conforme as métricas estatísticas extraídas. As subseções seguintes descrevem maiores de cada etapa.

4.1.1 *Medições de OWD*

O mecanismo SBD utiliza métricas estatísticas baseadas em medições de OWD para inferir se os subfluxos pertencentes a uma conexão de múltiplos caminhos estão compartilhando um gargalo entre si em um dado momento da transmissão. O OWD é determinado no nó receptor por

$$OWD = TS_r - TS_s, \tag{6}$$

sendo TS_r e TS_s são as marcações de tempo (*timestamp*) da recepção do pacote e do envio do mesmo pelo remetente, respectivamente. Embora os relógios dos nós emissor s e receptor r possam ter um *offset* e, portanto, estarem dessincronizados, isso não impacta a precisão do OWD. Nesse caso, para o mecanismo SBD não é importante observar os valores de OWD precisos em escala de tempo, mas sim capturar o comportamento e suas variações mediante ao tráfego corrente no enlace de gargalo. Assim, as métricas SBD são relativas à média dos OWDs medidos no receptor, de modo que o *offset* dos relógios se torna uma constante incorporada nas medições. Nesse método de medição, o OWD é denominado OWD relativo [46].

O mecanismo SBD, previsto na RFC 8382 [46], considera múltiplas janelas T de medições em um período de $M \leqslant N$ janelas. Portanto, a coleta de medições e extração

de estatística ocorrem sob o deslocamento de $M \times T$ janelas no tempo descritas na Tabela 4.

Tabela 4: Parâmetros de tempo com valores recomendados pela RFC 8382 [46].

Parâmetro	Valor	Descrição
T	350 ms	O intervalo de tempo base no qual as medições são feitas.
N	50	Número máximo de intervalos de tempo base, T, necessários para determinar métricas estatísticas.
M	30	Número mínimo de intervalos de tempo base, T, necessários para determinar métricas estatísticas, sendo $M \leqslant N$.

4.1.2 *Extração de Métricas Estatísticas*

4.1.2.1 *Atraso médio*

Dentro de um intervalo de tempo base T (350ms) várias medições de OWD podem ser amostradas. Dessa forma, o valor esperado, ou seja, o **valor médio de OWD no intervalo T** é dado por

$$E_T(OWD) = \frac{SUM_T(OWD)}{N_T(OWD)}, \tag{7}$$

onde $SUM_T(OWD)$ e $N_T(OWD)$ são a soma dos OWDs e o tamanho da amostra que foram observados em T. Em nossa implementação junto ao MPQUIC, ao final de um ciclo T, o valor esperado $E_T(OWD)$ é então armazenado em um *buffer* circular de $M \leqslant N$ elementos.

Dados os valores de $E_T(OWD)$ armazenados no *buffer*, determina-se a média global sobre o espaço M de valores. Dessa forma **o atraso médio** é dado por

$$MeanDelay = \frac{SUM_M(E_T(OWD))}{M}, \tag{8}$$

Em nossa implementação junto ao MPQUIC, considerando as primeiras medições, enquanto a quantidade de intervalos observados não atingir o tamanho $M = 30$, o atraso médio é relativo ao número de intervalos NUM_T observados até aquele momento.

4.1.2.2 *Estimativa de assimetria*

$SkewEst$ é uma estimativa da assimetria dos atrasos observados em um subfluxo durante um período de tempo. A assimetria é um valor que varia de positivo para negativo, quando se aproxima de 100% da carga suportada pelo caminho. Isso a torna uma métrica interessante, pois a ocorrência de congestionamento é justamente quando a carga excede a capacidade do enlace. Por outro lado, a assimetria torna-se positiva novamente em cargas acima de 125%, onde a taxa de perda excede 25% [47]. Dessa forma, subfluxos podem ser agrupados quando a assimetria é negativa ou a taxa de perda de pacote é alta.

A estimativa da assimetria é dada seguinte forma. Dado um OWD amostrado dentro de uma intervalo atual T, a sua assimetria pode ser determinada em relação ao atraso médio conhecido $MeanDelay$. Para tanto, a variável $SkewBase_T$ determina

de forma discreta essa assimetria, conforme o Algoritmo 2, sendo positiva quando a quantidade de observações OWD em T é menor que o atraso médio conhecido, ou negativo, caso contrário. Portanto, ao final de T, a variável $SkewBase_T$ fornecerá um indicador discreto do quantitativo da assimetria, a qual pôde ter variado positiva ou negativamente.

Algorithm 2 SkewBaseT: asymmetry in T

```
1: if OWD < MeanDelay then
2:     SkewBaseT += 1
3: end if
4: if OWD > MeanDelay then
5:     SkewBaseT -= 1
6: end if
```

Uma vez conhecida a assimetria do intervalo T, a estimativa de assimetria no intervalo $M \times T$ é dada pela média:

$$SkewEst = \frac{SUM_{M\times T}(skewBaseT)}{M}, \tag{9}$$

Em nossa implementação junto ao MPQUIC, ao final do intervalo T:

i) a variável skewBaseT é armazenada em um *buffer* específico de tamanho M e, em seguida, redefinida para zero;

ii) a média SkewEst é calculada ao longo do MxT. Assim como em MeanDelay, nos primeiros intervalos T a estimativa de SkewEst é relativa ao $M < 30$ conhecido.

4.1.2.3 *Estimativa de variabilidade*

VarEst é a estimativa de variabilidade dos atrasos observados pelo subfluxo durante um período de tempo. A variabilidade é uma função monotonicamente decrescente para os subfluxos que experimentam congestionamento. Portanto, isso a torna uma boa métrica para distinguir entre subfluxos que estão ou não experimentando um congestionamento [47].

A estimativa de variabilidade é dada por:

$$VarEst = \frac{SUM_{M\times T}(varBaseT)}{NUM_{M\times T}(OWD)}, \tag{10}$$

onde o numerado indica a soma no período $M \times T$ das variabilidades observadas em cada intervalo T, o denominador traz o total de medições de OWD realizadas no intervalo $M \times T$.

Para cada nova medição de OWD no intervalo T, o valor residual entre o OWD observado e o OWD médio conhecido é cumulativamente armazenado em VarBaseT. Portanto, a variabilidade observada em um intervalo T é dada por:

$$VarBaseT = SUM_T(|OWD - E_T(OWD)|), \tag{11}$$

onde $|\,x\,|$ é o valor absoluto de x, OWD é a nova medição de OWD, $E_T(OWD)$ é dado pela Eq. (7), i.e., a média calculada no intervalo T anterior.

Em nossa implementação junto ao MPQUIC, ao final de um intervalo T:

i) o valor de VarBaseT e a quantidade OWD medida no intervalo T são armazenados em um *buffer* específico de tamanho M;

ii) então, semelhante ao MeanDelay, o cálculo de VarEst considera os valores armazenados M × T ou é relativo nos primeiros intervalos T.

4.1.2.4 *Estimativa de oscilação*

FreqEst é a estimativa de uma oscilação de baixa frequência no atraso observado pelo subfluxo durante um período de tempo. Apenas uma medição consistente da frequência do subfluxo é capaz de distinguir subfluxos que estão ou não sob congestionamento [47].

A estimativa de oscilação de baixa frequência é determinada por:

$$\text{FreqEst} = \frac{\text{numberOfCrossings}}{N}, \tag{12}$$

onde numberOfCrossings armazena a contagem de oscilações do OWD médio durante a transmissão, e N é número máximo de intervalos de tempo base T.

Para determinar as oscilações, considera-se então a contagem de vezes em que o OWD médio conhecido foi maior ou menor que a amplitude média observada. Essa amplitude média é dada por:

$$\text{MeanDelay} \pm pv \times \text{VarEst}, \tag{13}$$

onde MeanDelay é o atraso médio conforme Eq.(8), pv é um limiar de sensibilidade que considera eventuais ruídos amostrais, e VarEst é a estimativa de variabilidade conforme Eq.(10).

As oscilações da média $E_T(OWD)$ então ocorrem a partir da alternância entre dois estados:

i) Quando $E_T(OWD)$ for alto a ponto de cruzar o limite superior da amplitude média;

ii) Ou quando $E_T(OWD)$ for baixo a ponto de cruzar o limite inferior da amplitude média.

Para cada nova média $E_T(OWD)$ conhecida, o número de oscilações em numberOfCrossings é determinado pela contagem de alternâncias entre dois estados, conforme descreve o Algoritmo 3.

Algorithm 3 FreqEst: Estimativa de oscilação.

```
if E_T(OWD) ⩾ (MeanDelay + pv × VarEst) and lastState ≠ 1 then
    numberOfCrossings += 1
    lastState = 1
else if E_T(OWD) ⩽ (MeanDelay − pv × VarEst) and lastState ≠ 2 then
    numberOfCrossings += 1
    lastState = 2
end if
```

Em nossa implementação junto ao MPQUIC, para o cálculo de *FreqEst*, definimos o denominador $N = M$. Assim como o atraso médio em $\mathtt{MeanDelay}$, a estimativa $\mathtt{FreqEst}$ é relativa enquanto o número de intervalos T for menor que $M = 30$.

4.1.2.5 *Taxa de perda de pacotes*

A taxa de perda de pacotes observada na janela $M \times T$ é dada por:

$$\mathtt{PktLoss} = \frac{\mathtt{SUM}_{M\times T}(\mathtt{lostPackets})}{\mathtt{SUM}_{M\times T}(\mathtt{totalPackets})}, \tag{14}$$

onde $\mathtt{lostPackets}$ é a quantidades de pacotes perdidos e $\mathtt{totalPackets}$ é o total de pacotes enviados em cada intervalo T.

Em nossa implementação junto ao MPQUIC, a quantidade de pacotes perdidos é detectada e enviada a cada pacote de dados ao receptor. O total de pacotes é computado no receptor pela quantidade de pacotes recebidos. Ao final de cada intervalo T, a variável de $\mathtt{PktLoss}$ é armazenada em um *buffer* específico de tamanho M.

4.1.3 *Agrupamento de Subfluxos*

Uma vez estimadas as métricas estatísticas no intervalo $M \times T$ para cada subfluxo pertencente a uma conexão de múltiplos caminhos no MPQUIC, inferem-se então os conjuntos de subfluxos que podem estar compartilhando gargalos em comum. O mecanismo previsto na RFC 8382 prevê cinco etapas para agrupar subfluxos congestionados, como ilustra a Figura 9, além de um conjunto de limiares, conforme descreve a Tabela 5. A subseções seguintes apresentam a operação do mecanismo em cada etapa.

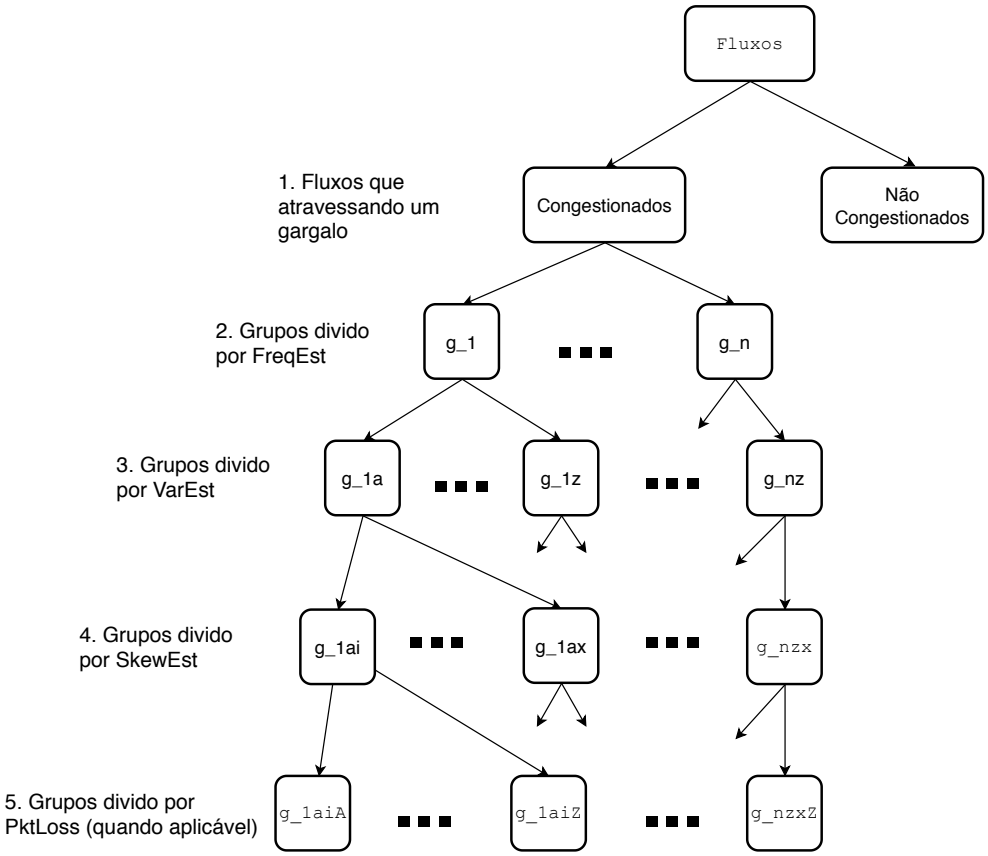


Figura 9: Diagrama do algoritmo de agrupamento
Fonte: imagem adaptada de [46].

Tabela 5: Limiares de utilizados pelo mecanismo de agrupamento de subfluxos previstos RFC 8382 [46].

Parâmetro	Valor	Descrição
pb	True, False	Variável booleana que indica se um determinado subfluxo foi ou não detectado como em trânsito em um gargalo no intervalo T anterior, ou seja, se percorreu um *gargalo anterior*.
cs	0.1	Limite na estimativa de assimetria (SkewEst) para controlar a sensibilidade do mecanismo na detecção de um gargalo.
pf	0.1	Limite na separação de subfluxo da métrica de estimativa de oscilação de baixa frequência (FreqEst).
pd	0.1	Limite na separação de subfluxo da métrica de taxa de perda de pacotes (PckLoss).
ps	0.15	Limite na separação de subfluxo da métrica de estimativa de assimetria (SkewEst).
pmad	0.1	Limite na separação de subfluxo da métrica de estimativa de variabilidade (VarEst).
pv	0.7	A sensibilidade da métrica de FreqEst ao ruído gerado pelas oscilações de latência ao longo do caminho.
ch	0.3	Limite da métrica de estimativa de assimetria (SkewEst) para controlar a histerese nos subfluxo
pl	0.1	Um limite para a métrica de taxa de perda de pacotes (PckLoss).

4.1.3.1 *Etapa 1: Agrupamento pela detecção de presença de congestionamento*

A RFC 8382 sugere três critérios para inferir se um dado subfluxo está congestionado, i.e., atravessa um gargalo:

$$(\mathtt{SkewEst} < \mathtt{cs}) \,||\, (\mathtt{SkewEst} < \mathtt{ch} \;\&\&\; \mathtt{pb}) \,||\, (\mathtt{PktLoss} > \mathtt{pl}).$$

O primeiro teste, $(\mathtt{SkewEst} < \mathtt{cs})$, considera a estimativa de assimetria do subfluxo em relação ao limiar $\mathtt{cs}$, o qual controla a sensibilidade do mecanismo. A parametrização por Hayes *et al.* [47] é $\mathtt{cs} = 0.0$. Um valor de $\mathtt{cs} = 0.1$ deixa o mecanismo um pouco mais sensível, enquanto $\mathtt{cs} = -0.1$ menos sensível. Quando ocorre um congestionamento, os valores medidos de OWD tendem a aumentar acima da média $\mathtt{MeanDelay}$. Consequentemente, os valores de $\mathtt{SkewbaseT}$ tendem a ficar menores, levando um indicativo para média $\mathtt{SkewEst}$ que o subfluxo medido está passando por um congestionamento. Em um determinado momento, os valores menores de $\mathtt{SkewEst}$ classificará o subfluxo como congestionado.

O segundo teste, $(\mathtt{SkewEst} < \mathtt{ch} \;\&\&\; \mathtt{pb})$, considera a estimativa de assimetria do subfluxo, contudo, em relação ao limiar de histerese $\mathtt{ch} = 0.3$ e também à *flag* $\mathtt{pb}$, que indica se subfluxo passou ou não por um gargalo no intervalo T_{i-1}. Esse critério verifica a histerese de subfluxos que foram agrupados sob compartilhamento de gargalo no intervalo anterior. Quando um subfluxo atravessa por um congestionamento, o valor de $\mathtt{MeanDelay}$ aumenta. Como o cálculo de $\mathtt{MeanDelay}$ considera os valores armazenados $M \times T$, dado um intervalo de T_i atual, os valores obtidos em T_{i-1} passados podem continuar influenciando na média. Se o valor de $\mathtt{MeanDelay}$ se tornar alto, e.g., devido a um tempo considerável que o subfluxo atravessa um congestionamento, os valores de OWD recentes podem ser menores que a média $\mathtt{MeanDelay}$, mesmo que o subfluxo esteja atravessando um congestionamento. Isso impactará uma sinalização em $\mathtt{SkewEst}$ que o subfluxo não atravessa um congestionamento, conforme o teste condicional anterior, levando a uma classificação de falso-negativo. Portanto, realiza-se esse segundo teste com um limiar ($\mathtt{cs}$) maior que o teste anterior, de modo a considerar o efeito de arrasto de $\mathtt{MeanDelay}$ sobre a métrica de $\mathtt{SkewEst}$ quando o subfluxo passou por congestionamento no intervalo anterior. O método é conservador no sentido de que, se for para errar a classificação, é preferível classificar um subfluxo como congestionado do que não congestionado. Dessa forma, uma classificação de falso-positivo pode levar a subutilização do subfluxo recentemente congestionado em T_{i-1}, mas não a introdução de mais carga sobre um gargalo que ainda pode estar congestionado em T_i.

O terceiro teste, $(\mathtt{PktLoss} > \mathtt{pl})$, considera a taxa de perda de pacotes e relação ao limiar $\mathtt{pl} = 0.1$. Quando um subfluxo está atravessando por um congestionamento, um dos indicativos é a perda de pacotes. Portanto, quando o $\mathtt{PktLoss}$ é alto, torna-se um indicador de congestionamento mais determinante que $\mathtt{SkewEst}$.

Caso ao menos um dos três critérios seja verdadeiro, o subfluxo então é marcado como transitante em gargalo através da sua *flag* $\mathtt{pb}$. Considerando que uma conexão de múltiplos caminhos possui um conjunto de um ou mais subfluxos ($\mathtt{subflows}$), esses critérios então devem ser verificados em cada subfluxo $\mathtt{sb}$ existente na conexão. Em nossa implementação no MPQUIC, essa verificação ocorre como descreve o Algoritmo 4, o qual é executado em cada ciclo T. Caso o subfluxo $\mathtt{sb}$ esteja percorrendo

um gargalo, além de pb, marca-se o seu atributo de compartilhamento (isSharing) como verdadeiro.

Algorithm 4 Detecção de gargalo nos subfluxos.

```
for sb = 1, 2, ... subflows do
    isCongested = False
    if (sb.SkewEst < cs) || (sb.SkewEst < ch && sb.pb) || (sb.PktLoss > pl) then
        isCongested = True
    end if
    sb.pb = isCongested
end for
```

Uma vez detectada a presença de gargalo nos subfluxos, a informação de compartilhamento dos gargalos então deve ser refinada. Nesse caso, o mecanismo prevê que esses subfluxos sob gargalos sejam progressivamente divididos em subgrupos, considerando os valores correntes de suas métricas estatísticas de FreqEst, VarEst e SkewEst. As subseções seguintes descrevem os critérios para realizar os agrupamentos sucessivos.

4.1.3.2 *Etapa 2: Agrupamento pela oscilação dos subfluxos sob gargalo*

Na segunda etapa de agrupamento, os subfluxos sob gargalos compartilhados são ordenados pela métrica de oscilação FreqEst. Esses subfluxos então são subdivididos pela diferença entre-valores ordenados de FreqEst. Para tanto, o critério de subdivisão é determinado pelo limiar de diferença pf da seguinte forma:

$$\mathtt{diff}(\mathtt{sort}(\mathtt{FreqEst})) < \mathtt{pf}.$$

Sendo x um *buffer* de comprimento M, a função $\mathtt{sort}(x)$ retorna o conjunto ordenado de forma crescente dos elementos em x, e a função $y = \mathtt{diff}(x)$ retorna um conjunto de comprimento $|y| = M - 1$, contendo as diferenças entre dois elementos adjacentes $\{x_2 - x_1, x_3 - x_2, \cdots, x_m - x_{m-1}\}$.

Valores similares de FreqEst indicam que os subfluxos atravessam gargalos com características similares, podendo, então, assumir que os subfluxos estão compartilhando gargalo. Portanto, ao ordenar os valores de FreqEst, valores similares se aproximam, e podendo assim ser agrupados juntos. Isto leva a considerar um valor de limiar pf sensível o suficiente para separação dos grupos que não apresentam valores similares.

Considerando um exemplo de $\mathtt{sort}(X) = \{11, 12, 13, 15, 16, 18, 19, 21\}$, a diferença seria $\mathtt{diff}(\mathtt{sort}(X)) = \{1, 1, 2, 1, 2, 1, 2\}$. Assumindo um limiar $\mathtt{pf} = 1.2$, tem-se a classificação dos subfluxos em quatro grupos distintos. Por simplificação, assumimos que $\mathtt{sort}(X) = Y$ e $\mathtt{diff}(\mathtt{sort}(X)) = Z$ e que os índices de Y corresponde ao ID de identificação de cada subfluxo. Em nossa implementação no MPQUIC, para realizar o agrupamento, primeiro cria-se um grupo com o valor do índice do primeiro elemento de Y. Em seguida, os elementos de Z serão testado ao limiar pf. Se o valor do primeiro elemento de Z for menor que o limiar, agrupa-se o valor do índice do segundo elemento de Y ao grupo anterior. Caso, o valor do elemento seguinte de Z for maior que o limiar pf, agrupa-se o valor do índice do terceiro elemento de Y em um novo grupo. E assim por diante. Esse exemplo é ilustrado na Figura 10.

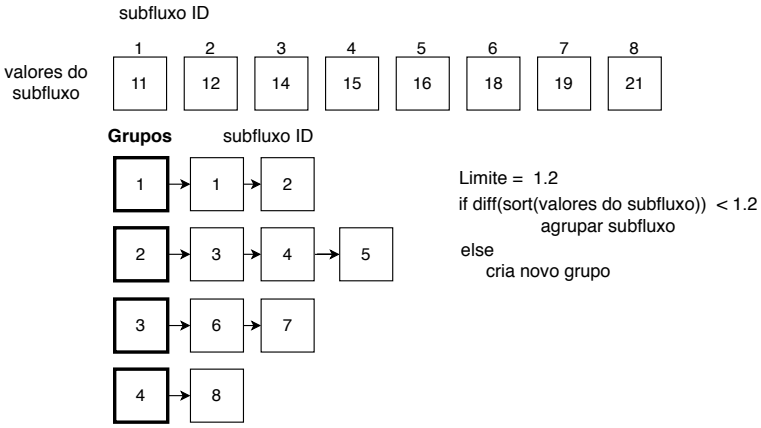


Figura 10: Exemplo de um agrupamento na etapa 2, considerando uma conexão que contenha 8 subfluxos.
Fonte: do autor

4.1.3.3 *Etapa 3: Agrupamento pela variabilidade dos subfluxos sob gargalo*

Na terceira etapa de agrupamento sucessivo, os grupos determinados na etapa 2 são subdivididos conforme as diferenças de variabilidades $E_M(VarEst)$ ordenadas do maior para o menor, em relação a um limiar de variabilidade. Conforme a RFC 8382, esse agrupamento é realizado pelo seguinte critério:

$$diff(sort(VarEst)) < (pmad \times VarEst),$$

onde o limiar $(pmad \times VarEst)$ se refere valor que causa a maior diferença, sendo $pmad = 0.1$. Da mesma forma como ocorre com $FreqEst$, valores similares de $VarEst$ indicam que os subfluxos estão compartilhando gargalo e, portanto, podem ser agrupados. Nesta etapa, o que muda é que o limiar é variável de acordo com $VarEst$, sendo 10% o valor do $VarEst$ que causou a maior diferença. O ponto onde causou a maior diferença, é um bom indicativo da separação entre dois grupos, caso exista. Portanto, o valor de $VarEst$ pode ser um bom estimador para o limiar de separação.

Em nossa implementação no MPQUIC, esse critério de agrupamento ocorre da seguinte forma:

$$diff(sort(VarEst)) < (pmad \times MaxVarEst),$$

onde $diff(x)$ é a diferença dos elementos adjacentes em x, $sort(x)$ é a ordenação crescente dos elementos em x, e $MaxVarEst$ é o valor referência para o $VarEst$ que causa a maior diferença. $MaxVarEst$ é determinado por

$$MaxDiff = max(diff(sort(VarEst))), \tag{15}$$

$$MaxVarEst = GetVarEst(MaxDiff), \tag{16}$$

onde $MaxDiff$ é a maior diferença entre as métricas de $VarEst$ ordenadas de forma decrescente e $GetVarEst(x)$ sendo o respectivo valor de $VarEst$ o qual causou $MaxDiff$. Supondo $sortAZ(x) = \{1, 2, 5, 6, 7\}$, então $MaxDiff = 3$ e, portanto, $MaxVarEst = 5$.

4.1.3.4 *Etapa 4: Agrupamento pela assimetria dos subfluxos sob gargalo*

Na quarta etapa de agrupamento sucessivo, os grupos de subfluxos determinados pelo critério de variabilidade são subdivididos pelo critério de assimetria através da métrica $\mathtt{SkewEst}$. Esse agrupamento é realizado de forma similar aos das etapas 2 e 3, considerando, desta vez, as diferenças entre os valores adjacentes de assimetria:

$$\mathtt{diff}(\mathtt{sort}(\mathtt{SkewEst})) < \mathtt{ps},$$

onde $\mathtt{ps} = 0.15$ é o limiar considerado para as diferenças entre as assimetrias adjacentes. Da mesma forma como ocorre nas etapas anteriores, valores similares de $\mathtt{SkewEst}$ indicam que os subfluxos podem ser agrupados. Isto leva a considerar um valor de limiar $\mathtt{ps}$ sensível o suficiente para separar em subgrupos os subfluxos similares.

4.1.3.5 *Etapa 5: Agrupamento pela perda de pacotes dos subfluxos sob gargalo*

Finalmente, na etapa 5, os subfluxos agrupados pela assimetria na etapa 4 são subdivididos pela taxa de perda de pacotes $\mathtt{PktLoss}$. Esse agrupamento final considera as diferenças entre os valores adjacentes de perdas:

$$\mathtt{diff}(\mathtt{sort}(\mathtt{PktLoss})) < (\mathtt{pd} \times \mathtt{PktLoss}),$$

onde o limiar de $(\mathtt{pd} \times \mathtt{PktLoss})$, sendo $\mathtt{pd} = 0.1$, é referente ao maior valor da diferença, o qual é determinado de forma similar ao $\mathtt{MaxVarEst}$ na etapa 3. Assim como nas etapas anteriores, os subfluxos são agrupados pela similaridade, desta vez, considerando as diferenças ordenadas de taxas em relação a um limiar de perda. A maior diferença entre as taxas de $\mathtt{PktLoss}$ é um ponto de ruptura entre dois grupos. Portanto, o maior valor de diferença em $\mathtt{PktLoss}$ pode ser um bom estimador para o limiar de separação.

4.2 IMPLEMENTAÇÃO NO MPQUIC

Como descrito anteriormente, QUIC é um protocolo de camada de aplicação, sendo o MPQUIC uma evolução do QUIC que possibilita transmissão por múltiplos caminhos. Enquanto há diversas implementações do protocolo QUIC em diferentes linguagens de programação [99], o MPQUIC está disponível em poucas implementações. Considerando que o foco deste trabalho de mestrado é implementar e validar um mecanismo de SBD (*Shared Bottleneck Detection*) baseado na RFC 8382 junto ao protocolo MPQUIC, a implementação do MPQUIC a qual utilizamos como ponto de partida é `mpquic-go` de De Coninck e Bonaventure [21]. Essa foi primeira implementação do MPQUIC a ser proposta, a qual se deu a partir da implementação do protocolo QUIC na linguagem Go (`quic-go`) [116]. Portanto, o mecanismo SBD está sendo implementado em linguagem Go, estendendo a implementação de `mpquic-go` [21]. As subseções seguintes descrevem pontos importantes da implementação do MPQUIC-SBD, em termos de componentes da arquitetura, principais procedimentos e estrutura dos pacotes.

4.2.1 *Considerações de Projeto*

Como as métricas estatísticas baseiam-se nas amostras de OWD coletadas pelo receptor, enquanto o emissor contabiliza as perdas de pacotes, não é possível implementar o mecanismo SBD de forma autônoma, complemente no emissor ou completamente no receptor. Dessa forma, tem-se ao menos duas diferentes abordagens para implementação do mecanismo:

(1) **No emissor**. Requer que o receptor retorne ao emissor via pacote ACK os valores de OWD medidos, de modo que o emissor execute as operações de extração de métricas e agrupamentos dos subfluxos.

(2) **No receptor**. Além das medições de OWD, o receptor também é responsável por extrair as métricas e realizar o agrupamento dos subfluxos, retornando ao emissor via pacotes ACK somente o resultado do mecanismo, i.e., a informação de agrupamento dos subfluxos.

Independente da abordagem, as decisões que o emissor (controle de congestionamento e escalonador de pacotes) poderá tomar com base no suporte do mecanismo SBD sofrerá ao menos um atraso 1RTT/2. Ambos pares precisam de informações um do outro para prosseguirem tanto no mecanismo SBD ou quanto na tomada de decisão. Entretanto, mesmo diante dessa possibilidade, entendemos que a abordagem (2) é mais vantajosa. Nesse caso, o mecanismo é mais ágil, pois as entradas para o mecanismo estão disponíveis tão logo o receptor realizar as medições de OWD. Além disso, o receptor pode executar quase todas as operações do mecanismo de forma independente, exceto sobre a perda de pacotes. Outra vantagem é que se evita maior transmissão de dados importantes de controle, como amostras de OWD e métricas, via pacotes ACK, os quais podem ser perdidos e, nesse caso, não são retransmitidos.

Uma sessão MPQUIC entre o emissor, receptor e seus principais componentes são ilustrados na Figura 11. A figura ilustra uma sessão MPQUIC de quatro subfluxos. O processamento de pacotes por múltiplos caminhos requer que o MPQUIC tenha determinados componentes específicos por subfluxo, como os *handlers* de envio e recebimento, controle de congestionamento. No escopo da sessão, o Gerenciador de Caminhos é responsável por lidar com o estabelecimento e monitoramento dos subfluxos. O Controle de Congestionamento dos subfluxos operam de forma acoplada, considerando a agregação das capacidades de todos os subfluxos para então ajustar dinamicamente as janelas individuais. Quando um pacote está pronto para ser transmitido, o escalonador de pacotes é responsável por determinar qual subfluxo irá enviá-lo. O mecanismo SBD é um novo módulo acoplado ao gerenciador de caminhos no receptor, o qual mede OWDs e extrai as métricas estatísticas de cada subfluxo para, então, agrupar os subfluxos sob gargalos compartilhados, conforme descrito na Seção 4.1.

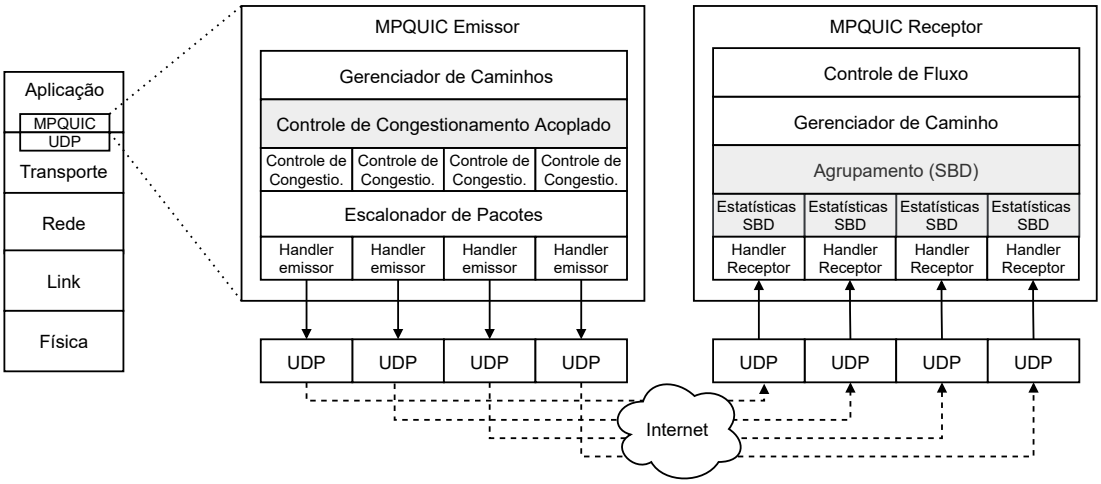


Figura 11: Componentes de uma implementação do MPQUIC e sua modificação (cinza) com o mecanismo SBD, considerando a abordagem de implementação mais complexa no receptor.
Fonte: imagem adaptada de Viernickel *et al.* [125]

4.2.2 *Novos Campos nos Quadros*

Tendo em vista que a abordagem escolhida foi a implementação de um receptor MPQUIC mais complexo, foi necessário modificar a estrutura tanto de pacotes `ACK` quanto de `stream`. A ilustração na Figura 12 mostra as modificações.

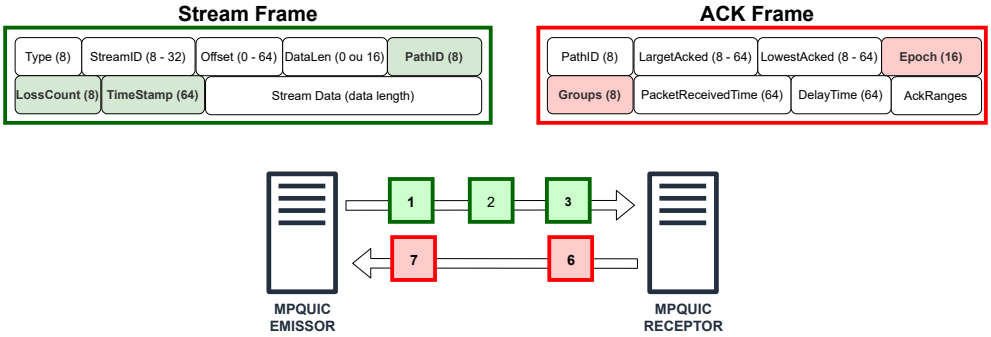


Figura 12: Emissor, receptor e seus quadros de fluxo de dados (*stream*) e reconhecimento (*ACK*) transmitidos em um subfluxo. Novo campos incluídos pelo mecanismo SBD são `PathID`, `LossCount`, `TimeStamp`, `Epoch` e `Groups` nos quadros de dados e de reconhecimento, respectivamente. Comprimento dos campos em bits.
Fonte: do autor.

Nos pacotes de confirmação, agora há dois novos atributos chamados `Epoch` e `Groups`, de 2 e 1 byte de comprimento cada, respetivamente, o qual informa ao emissor a saída do mecanismo SBD, contendo a identificação da época e do grupo o qual o subfluxo pertencente naquele momento. Como a classificação do grupo é enviada individualmente via ACK, pode chegar de forma não sincronizada, portanto, mantém-se uma identificação da classificação daquele grupo, sendo chamada de época, evitando inconsistência ao comparar grupos de classificação diferentes. Um espaço de 1 byte permite a identificação de até 256 grupos, portanto, é bastante suficiente para

a finalidade de agrupamento de subfluxos no MPQUIC. O quadro *Ack Frame* contém o campo `PathID` que indica qual caminho o quadro pertence. Portanto, a informação da classificação de cada caminho é armazenada em um *buffer* e inserida no *Ack Frame* correspondente. O mecanismo é redundante, enviando a mesma informação de agrupamento atual em todos os quadros ACK do receptor para o emissor, e não somente quando uma nova configuração de agrupamento é determinada. Esse mecanismo mitiga a possibilidade de perda de informação, caso quadros ACK sejam perdidos.

Como o mecanismo de depende das informações de pacotes perdidos e da marcação do tempo de envio do pacote, as quais são determinadas pelo emissor, os pacotes de *stream* utilizados pelo emissor na transmissão de dados das aplicações tem seu cabeçalho modificado para conter os novos atributos: `LossCount` (1 Byte), `TimeStamp` (8 bytes) e `PathID` (1 Byte). A partir do `LossCount` que contém a quantidade de pacotes perdidos até o momento, o receptor pode calcular a taxa de `Pktloss`. O atributo `TimeStamp` informa ao receptor o tempo de envio do pacote, para que o mesmo possa medir o OWD tão logo receber o pacote. O `PathID` é importante para que o receptor distingua os caminhos, os quais os atributos recebidos pertencem.

4.2.3 *Principais Procedimentos no Receptor*

O mecanismo SBD é executado conforme o fluxograma na Figura 13, a partir das principais procedimentos. Os demais procedimentos e atributos podem ser verificados no diagrama de classes da implementação do MPQUIC-SBD na Figura 14. Como mencionado anteriormente, as métricas são computadas no nó receptor, juntamente com a informação de pacotes perdidos pelo emissor até o momento. Para cada novo pacote i recebido, determina-se a medição de OWD pelo tempo decorrido do recebimento do pacote anterior $i-1$. A cada novo pacote recebido, faz-se então a diferença do tempo de chegada desse pacote com o imediatamente anterior.

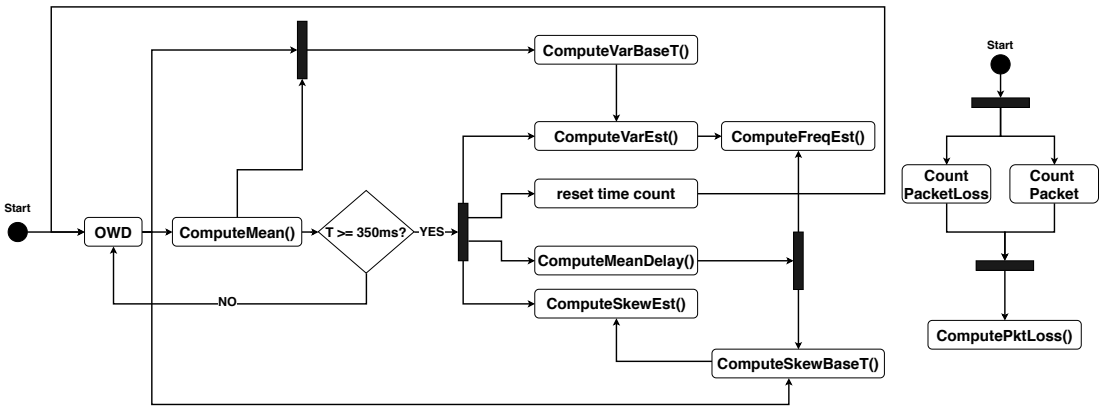


Figura 13: Fluxograma da operação do mecanismo SBD junto ao MPQUIC.
Fonte: do autor.

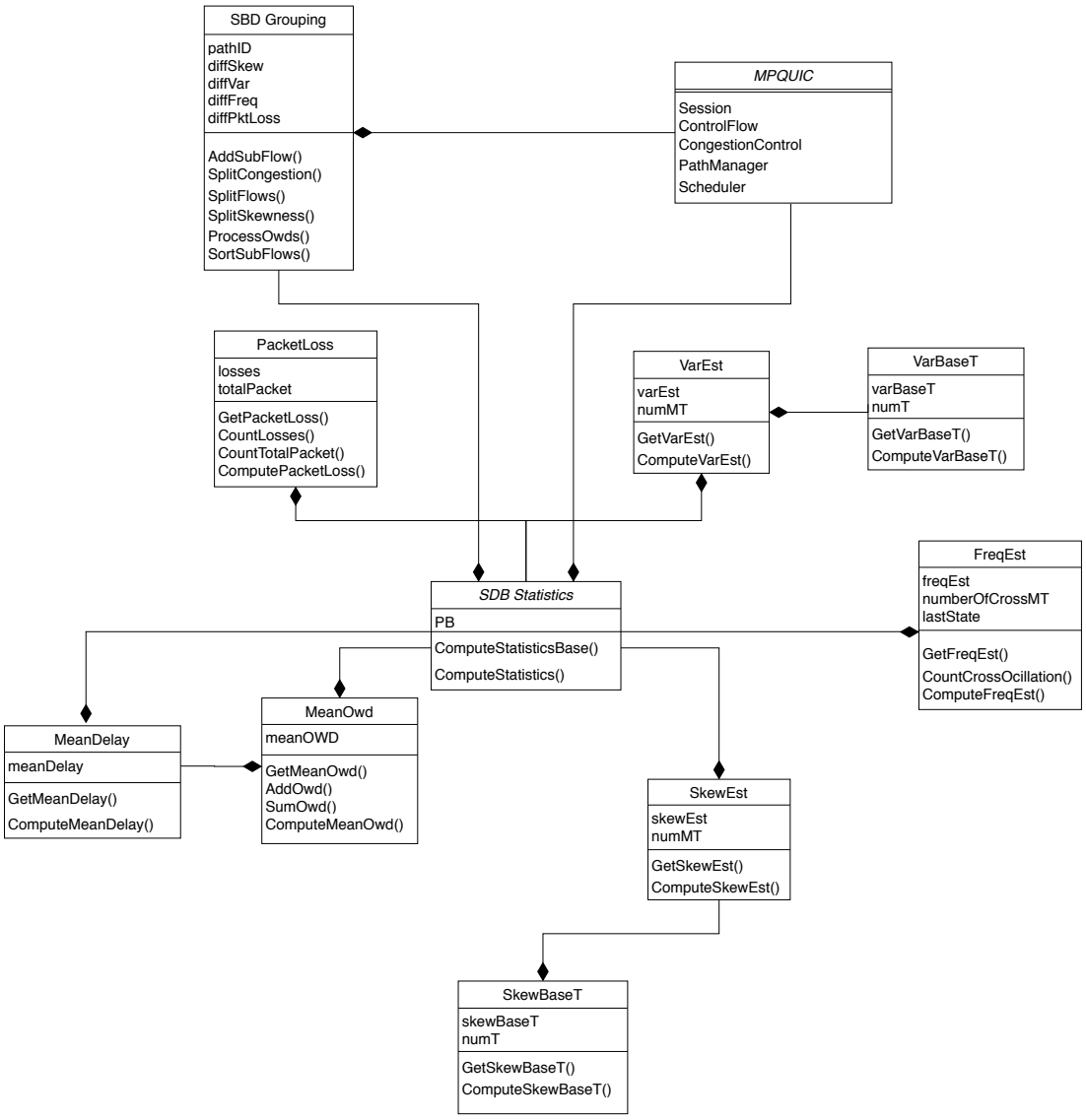


Figura 14: Diagrama de Classes da implementação MPQUIC-SBD.
Fonte: do autor.

Com a medição mais recente de OWD, então:

- Atualiza-se à média $E_T(OWD)$ com o procedimento `ComputeMean()`, conforme Equação (7),
- Soma-se à $VarBaseT$ o valor residual entre o OWD observado e a média conforme Equação (11).
- Computa de forma discreta a assimetria em $SkewBaseT$, conforme Algoritmo 2.

Como prevê a RFC 8382, as métricas são obtidas em janelas de tempo $M \times T$, tendo $M = 30$ e $T = 350$ ms. Dessa forma, dentro da janela $T < 350$ ms, apenas as métricas básicas de média $E_T(OWD)$ e $VarBaseT$ são calculadas. A cada término do intervalo T, reinicializa-se o contador e atualizam-se as métricas $MeanDelay$, $SkewEst$, $VarEst$ e $FreqEst$, conforme Equações (8), (9), (10) e (12), implementadas através dos procedimentos `ComputeMeanDelay()`, `ComputeSkewEst()`, `ComputeVarEst()` e `ComputeFreqEst()`, respectivamente.

4.2.4 *Controle de Congestionamento baseado em SBD no Emissor*

Para cada nova classificação de agrupamento, cada subfluxo receberá uma classificação de grupo e época ao qual pertence. Como o receptor irá enviar de forma individual a classificação via ACK, o emissor irá manter um conjunto de histórico das épocas recebidas, quando todas os subfluxos forem atualizados, a época se torna válida, isto é feito para que não haja inconsistência nos dados, ao realizar os cálculos com classificação de grupos em épocas diferentes.

Essencialmente, o objetivo do método SBD é prover mais informação ao emissor para que o mesmo tome melhores decisões de transmissão de pacotes. Essas decisões podem ser tomadas, principalmente, no controle de congestionamento dos subfluxos e/ou no escalonamento de pacotes sobre os subfluxos. Ao passo que decisão de escalonamento ocorre para o pacote pronto para ser transmitido, no controle de congestionamento a decisão ocorre no nível da carga de trabalho do subfluxo.

Se dois subfluxos estão atravessando condições de tráfego intenso, mas em caminhos diferentes, um acaba influenciando no cálculo da janela de congestionamento do outro, pois o controle acopla todos os subfluxos para atualizar as janelas de congestionamento. Entretanto, os subfluxos percorrem caminhos disjuntos, i.e., gargalos diferentes, de modo que, na prática, a carga de trabalho de um não impacta na do outro. Dessa forma, optamos por modificar o controle de congestionamento padrão do protocolo MPQUIC, sendo implementado através do algoritmo OLIA [68] (Subseção 3.2.2).

Em nossa implementação, denominada **OLIA-SBD**, ao prover a classificação dos subfluxos que estão ou não compartilhando gargalo, o mecanismo SBD permite ao emissor um melhor controle de congestionamento:

- **Acoplar** o controle quando os subfluxos estiverem compartilhando um gargalo em comum;
- **Desacoplar** o controle quando os subfluxos não estiverem compartilhando gargalo.

Mais detalhadamente, a operação em OLIA leva em consideração três condições para acoplar ou não o cálculo da janela de congestionamento de um subfluxo r aos demais subfluxos disponíveis:

(1) O subfluxo não está congestionado.

(2) O subfluxo pertence ao mesmo grupo de r.

(3) O subfluxo pertence a um grupo diferente de r.

Na condição (1) o subfluxo não teve uma classificação SBD, pois, não está congestionado, portanto, assume-se a operação padrão em OLIA e inclui-se o subfluxo no cálculo acoplado. Na condição (2), o subfluxo pertence ao mesmo grupo de r, portanto, compartilha um mesmo gargalo, inclui-se no cálculo acoplado. Na condição (3), o subfluxo não pertence ao mesmo grupo, i.e., não compartilha gargalo, portanto não o incluí no cálculo acoplado. O Algoritmo 5 apresenta a função que recebe r e retorna os estados de cada subfluxo, para verificar quais subfluxos estão acoplados ou não a r. Portanto, o algoritmo busca entre todos os subfluxos identificando quais

subfluxos pertence ao mesmo grupo do subfluxo avaliado, se encontrar retornará um vetor com as identificações dos subfluxos pertencente ao mesmo grupo e os quais não pertecem a nenhum grupo.

Algorithm 5 Verifica os subfluxos que pertencem ao cálculo acoplado da janela de congestionamento

```
function PertenceAoCalculoAcoplado(r)
    epoch = getEpoch(r)              ▷ Busca a última época válida para o subfluxo analisado
    group = getGroup(r, epoch)       ▷ Busca o grupo da época ao qual o subfluxo pertence
    for x = 1, 2, ... subflows do
        pertence[i] = false
        skGroup = getGroup(x, epoch)
        if (skGroup == 0) then
            pertence[i] = true
        else if (skGroup == group) then
            pertence[i] = true
        end if
    end for
    return pertence
end function
```

A seção seguinte discute os principais trabalhos relacionados.

4.3 TRABALHOS RELACIONADOS

Os trabalhos iniciais de Savage *et al.* [115] consideram o compartilhamento de informação entre sistemas finais para aprimoramento do controle de congestionamento. Os autores sugerem esquemas de monitoramento do tráfego passivo para obter informações de atrasos e perdas, além de comparar endereços IP para determinar as localidades dos nós na rede. Por conta de poucas redes que são alocadas em uma granularidade menor que 256 nós, as informações obtidas não são suficientes para determinar uma boa estimativa da localidade dos nós. Portanto, os autores ressaltam não ser suficiente para uma proposta de um novo protocolo de transporte.

Katabi *et al.* [66] desenvolveram uma abordagem passiva para detectar gargalos compartilhados em uma rede, cuja topologia é definida em forma de árvore. Os caminhos que compartilham um mesmo roteador intermediário são considerados caminhos de agrupamentos. Dada a observação de um nó comum, pode-se minimizar a entropia do espaçamento entre pacotes visto pelo observador, assim recuperando o espaçamento constante entre os pacotes quando foram inseridos na rede. Podendo, então, descobrir os fluxos que compartilham o gargalo. No entanto, a precisão desse observador completamente passivo diminui quando se observa apenas uma pequena fração do tráfego no gargalo.

O trabalho de Rubenstein *et al.* [111] apresenta as primeiras investigações sobre a detecção de gargalos compartilhados. Os autores propõem a medição de correlação de perda e atraso entre pares de pacotes, relacionando a correlação cruzada com a correlação entre pares dentro do fluxo. Caso a correlação cruzada seja maior que a correlação dentro do fluxo, conclui-se que os fluxos compartilham gargalos. Os autores identificaram que o método com correlação por atraso converge mais rapidamente para resposta correta, em comparação à correlação baseado em perdas de

pacotes. Além disso, pode não haver amostras suficientes para que a correlação de perdas seja capaz de concluir se os subfluxos estão compartilhando gargalo.

Hassayoun *et al.* [45] propuseram DWC (*Dynamic Window Coupling*), mecanismo o qual acopla dinamicamente as janelas de congestionamento de seus subfluxos em uma conexão MPTCP. O agrupamento dos subfluxos considera medições de perdas e atrasos de ida e volta (RTT) observados. Nesse caso, quando ocorre uma perda ou um aumento nos valores de RTT próximo a um evento de perda, os sub-fluxos são classificados como em estado de congestionamento compartilhando um gargalo. Isso é realizado em todos os eventos de congestionamento que resultem em perda de pacotes. A cada evento, reagrupam-se os sub-fluxos, garantindo capacidade de resposta ao tráfego ou à alterações de rota. Assim, limitando o impacto de uma detecção errada. Bem como adaptando às mudanças da rede e se recuperando rapidamente de erros para a próxima evento de congestionamento. Com DWC, obteve-se maior taxa de transmissão MPTCP, especialmente quando os caminhos usados pelos sub-fluxos possuem gargalos disjuntos, obtendo também um melhor desempenho em sub-fluxos que compartilham um gargalo.

Zhu [134] propôs um método de medição ativa para identificar fluxos que compartilham gargalo. Sondas são injetadas nos tráfegos para medir o intervalo de tempo dos pacotes nas extremidades receptoras. Uma transformação de Fourier com o sinal de atraso de chegada entre pacotes injetados no tráfego é usada para detectar se os fluxos TCP compartilham gargalo.

Hayes *et al.* [47] propuseram o método SBD (*Shared Bottleneck Detection*), para detecção de gargalos compartilhados, recorrendo à análise estatística de fluxos fim-a-fim. A análise baseia-se na coleta de tempos de ida OWD (*One-Way Delay*) e, com base em conjuntos de OWD amostrados, determinam-se métricas estatísticas de resumo das transmissões nos domínios de tempo e frequência. As métricas agrupam subfluxos a partir das diferenças estatísticas de pares de fluxo para, então, determinar se estão compartilhando gargalo. O método proposto se mostrou robusto em experimentos tanto em simulações quanto em ambientes reais. O método possui uma ampla variedade de aplicação, desde aplicações WebRTC Alvestrand [5] até possível implementação na pilha de protocolos no núcleo de sistemas operacionais. Em [31], os autores aplicam o método SBD junto ao controle de congestionamento do protocolo MPTCP, e para o cenário de gargalo não compartilhado observando ganhos de até 40% na taxa de transferência com dois sub-fluxos e os ganhos aumentam significativamente à medida aumenta o número de sub-fluxos, atingindo mais de 100% durante cinco sub-fluxos. Além disso, para o cenário de gargalo compartilhado, os autores mostraram que o MPTCP-SBD permanece justo com fluxos TCP.

Wei *et al.* [128] propuseram SBDV (*Shared Bottleneck Detection Based on Congestion Interval Variance Measurement*), método que prevê a medição da variância de OWDs para detectar gargalos compartilhados. O objetivo é melhorar a acurácia de medições de fluxos que compartilham gargalos em comum, mas, por percorrerem caminhos distintos, apresentam latências diferentes – essas situações são denominadas *path lag* na literatura. Para tanto, SBDV opera da seguinte forma. Primeiro, define-se um delta $\Delta = t_2 - t_1 = (t_1 - t_0) - (t_2 - t_0)$, sendo t_1 e t_2 os tempos de chegada no receptor dos fluxos 1 e 2, respectivamente, correspondendo aos atrasos dos enlaces P1 e P2. Com isso, define-se a variância em relação ao delta, $Vari(\Delta)$, sendo um valor que reflete a intensidade da flutuação do atraso na fila nos enlaces não congestionados, ou

seja, P1 e P2. Gargalos compartilhados são detectados entre P1 e P2 quando $Vari(\Delta)$ ultrapassa um limiar, sendo a média quadrática dos tempos de duração do evento de congestionamento dos dois subfluxos que estão compartilhando gargalo. O algoritmo demonstrou robustez e eficácia em cenários de *path lag*. Em um outro trabalho [129], os autores abordam o problema de forma diferente, utilizando o suporte do mecanismo ECN (*Explicit Congestion Notification*) para detectar gargalos compartilhados e estimar o grau de congestionamento sobre cada subfluxo. Essa detecção é aplicada nos mecanismos de controle de congestionamento e escalonamento de pacotes, denominados SB-CC e SB-FBS, respectivamente, para melhorar a vazão de transmissões MPTCP. SB-CC equilibra as cargas nos subfluxos e suaviza a flutuação da janela de congestionamento. Para evitar a degradação da taxa de transmissão por devido à chegada de pacotes fora de ordem, SB-FBS distribuir os pacotes conforme as alterações nas janelas de congestionamento, e, portanto, pode escalonar com maior precisão os pacotes em cenários de gargalo compartilhado.

Este trabalho parte dos estudos de Hayes *et al.* [47] e Ferlin *et al.* [31], os quais levaram a definição do padrão RFC 8382 [46], para a implementação e validação de um mecanismo SBD junto ao protocolo MPQUIC. Embora existam diferentes abordagens para detectar compartilhamento gargalos, o padrão RFC 8382 representa o estado da arte. No presente momento, este trabalho é o primeiro a propor, implementar, validar e discutir resultados experimentais do suporte SBD junto ao protocolo MPQUIC. A Tabela 6 apresenta uma comparação qualitativa dos principais trabalhos relacionados, considerando suas métricas, métodos, objetivos e protocolos aplicados.

Tabela 6: Comparação entre os principais trabalhos relacionados.

Trabalhos	**Métrica**	**Método**	**Objetivo**	**Protocolo**
Savage *et al.* [115]	Estatísticas de atraso e perda	Passivo	ACC	TCP
Katabi *et al.* [66]	Entropia do espaçamento entre pacotes	Passivo	DGC	TCP/UDP
Rubenstein *et al.* [111]	Correlação de perda e atraso	Passivo	DGC	TCP
Hassayoun *et al.* [45]	Estatísticas de atraso e perda	Passivo	ACC	MPTCP
Zhu [134]	Transformada de Fourier sobre atraso	Ativo	DGC	TCP
Hayes *et al.* [47]	Estatísticas de OWD	Passivo	ACC	MPTCP
Ferlin *et al.* [31]	Estatísticas de OWD	Passivo	ACC	MPTCP
Wei *et al.* [128]	Variância de OWD	Passivo	ACC	MPTCP
Wei *et al.* [129]	Notificações ECN	Passivo	ACC e AEP	MPTCP
***Este trabalho**	**Estatísticas de OWD - RFC 8382 [46]**	**Passivo**	**DGC e ACC**	**MPQUIC**

Legenda: DGC Detecção de Gargalos Compartilhados
ACC Aprimoramento de Controle de Congestionamento
AEP Aprimoramento de Escalonamento de Pacote

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou maiores detalhes do desenvolvimento de um mecanismo de detecção de gargalos compartilhados (SBD) baseado na RFC 8382 [46] junto ao protocolo MPQUIC. Tal desenvolvimento vem sendo feito a partir da implementação do MPQUIC em Linguagem Go (`mpquic-go`) disponibilizada por De Coninck e Bo-

naventure [21]. Uma vez que o mecanismo SBD é implementado como componente de software, um módulo quase independente, sua implantação junto ao `mpquic-go` tem sido simples, não exigindo grandes alterações enquanto preserva a estrutura da implementação do protocolo.

Essa nova versão do protocolo, o qual denominamos MPQUIC-SBD, é objeto de investigação neste trabalho de mestrado. Não somente os aspectos tecnológicos de implementação do mecanismo SBD são estudados, mas também a identificação dos impactos da aplicação da informação de SBD disponibilizada pelo mecanismo. Com MPQUIC-SBD, tais impactos poderão ser investigados experimentalmente em transmissões diversas, especialmente as de multimídia, considerando tomadas de decisão baseada em SBD junto ao controle de congestionamento e escalonamento de pacotes no emissor. Além disso, a implementação da RFC 8382 em MPQUIC-SBD poderá servir de base para comparações com futuros algoritmos e métodos de detecção de compartilhamento de gargalos em transmissões MPQUIC.

5 RESULTADOS EXPERIMENTAIS

Para avaliar o desempenho do MPQUIC-SBD, foi considerada a aplicação de transmissão de vídeo. Essa aplicação é particularmente interessante, pois as transmissões de video atualmente se destacam como o tipo de tráfego mais realizado por usuários de *smartphones*, em uma média atual de 60% do tráfego total, com aumento projetado para 74% do tráfego no final de 2024 [28]. Além disso, as tomadas de decisões de emissores equipados com mecanismos SBD nessas aplicações ainda é pouco conhecida, sobretudo quando as transmissões de vídeo são realizadas a partir de um protocolo recente, como MPQUIC.

Este capítulo apresenta o seguinte conteúdo. Detalhes são apresentados sobre as soluções avaliadas, tanto da aplicação de vídeo quanto dos mecanismos de transporte MPQUIC. Os cenários de rede implementados em emulador, com as configurações de gargalos e tráfego de fundo, são descritos. Os resultados de experimentos conduzidos em emulador são discutidos, considerando: a acurácia do algoritmo de agrupamento SBD; a Qualidade de Serviço observada no transporte MPQUIC, em termos de vazão e latência de transmissão de segmentos de vídeo; a Qualidade de Experiência observada na aplicação de vídeo a partir de diferentes indicadores. Finalmente, o capítulo se encerra com considerações finais.

5.1 SOLUÇÕES AVALIADAS

A Figura 15 ilustra a pilha de protocolos e a implementação das soluções avaliadas nos experimentos discutidos neste capítulo. O nó MPQUIC foi equipado com duas interfaces de rede (NIC), considerando diversidade mais típica de multiconexão (*multi-homing*) na Internet presente em dispositivos de usuário, como *smartphones*.

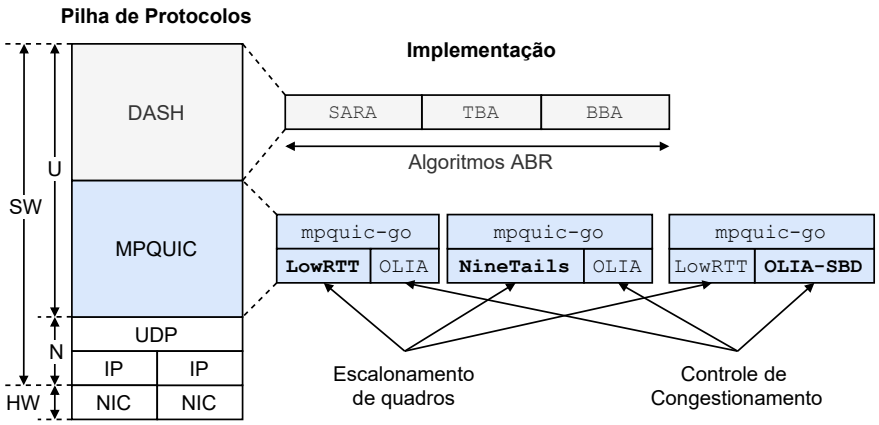


Figura 15: Pilha de protocolos e implementação das soluções avaliadas nas camadas DASH e MPQUIC. Legenda: U (espaço de usuário), N (espaço de núcleo), SW (software), HW (hardware), NIC (*network interface controller*).

Na camada de aplicação, as transmissões de vídeo ocorrem através de DASH (*Dynamic Adaptive Streaming over HTTP*), o qual é de fato o padrão de *streaming* de video atual da Internet [118]. Três algoritmos ABR (*Adaptive Bit Rate*) populares em DASH são avaliados: SARA, TBA e BBA. Ainda na camada de aplicação, contudo, abaixo de DASH, estão as soluções de transporte MPQUIC: LowRTT, NineTails e OLIA-SBD, conforme descreve a Tabela 7.

Tabela 7: Soluções MPQUIC utilizadas avaliadas nos experimentos.

Soluções	**Descrição**
NineTails	É um escalonador redundante [126]. Diferentemente, para outros algoritmos redundantes, que simplesmente replicam quadros em todos os subfluxos existentes, o NineTails é seletivo ao usar o modo de redundância apenas próximo ao final das mensagens. A lógica por trás disso é que as perdas que ocorrem nos últimos quadros causam um aumento significativo da latência devido ao RTO, enquanto latência extra insignificante é observada em perdas recuperadas (*fast retransmissions* em meio às transmissões das janelas de congestionamento em andamento) antes dos últimos quadros. Nesta composição de soluções MPQUIC, NineTails é o escalonador e OLIA é controle de congestionamento padrão (Subseção 3.2.2).
LowRTT	Representa a composição de soluções padrão do MPQUIC, ou seja, escalonador LowRTT e controle de congestionamento OLIA (Subseção 3.2.2).
SBD	Controle de congestionamento com suporte a SBD proposto, OLIA-SBD (Subseção 4.2.4). A classificação dos grupos SBD para cada subfluxo irá acoplar ou desacoplar os subfluxos durante os cálculos do aumento da janela de congestionamento em OLIA-SBD. Acopla-se, ao detectar que subfluxos compartilham um mesmo gargalo; desacopla-se, caso contrário. Nesta composição de soluções MPQUIC, utiliza-se o escalonador padrão LowRTT (Subseção 3.2.2) e OLIA-SBD como controle de congestionamento.

5.2 APLICAÇÃO DE VIDEO

Na aplicação, as transmissões MPQUIC ocorrem entre um cliente DASH e um servidor HTTP, o qual armazena diferentes segmentos de um vídeo. As subseções seguintes descrevem detalhes.

5.2.1 *Implementação da Aplicação*

O cliente DASH é implementado pelo *player* `AStream`[1] [63] e o servidor HTTP pelo `Caddy`[2]. Entre os detalhes da implementação da aplicação de vídeo, enquanto um *player* DASH é capaz de adaptar as taxas de bits dos segmentos em tempo de reprodução do vídeo, um servidor disponibiliza codificações do vídeo em diferentes representações que variam em taxas de bits, resolução de exibição, etc. Além disso, cada representação é dividida em vários pedaços de vídeo menores de duração fixa, denominados segmentos, por exemplo, de 1, 2, 4 ou 10 segundos. No lado do cliente,

1 https://github.com/pari685/AStream
2 https://caddyserver.com

o *player* DASH, através de um algoritmo do tipo ABR (*Adaptive Bit Rate*), determina a representação apropriada para cada segmento a ser requisitado ao servidor, mediante a algum critério, e.g., condições da rede, como latência e taxa de transmissão atual, espaço em *buffer*, tamanho do segmento.

O vídeo utilizado nos experimentos possui duração de 3m59s, o qual foi dividido em segmentos de 4 segundos – comprimento típico de segmento em aplicações DASH [63, 64]. O servidor HTTP armazena segmentos de vídeo de diferentes qualidades, variando de 90 a 2560 pixels sobre taxas de bits (*bitrate*) de 250 Kbps a 20000 Kbps. Dessa forma, os algoritmos ABR na camada DASH podem variar as requisições de segmentos em um espaço amplo de segmentos disponível no servidor.

O processo de avaliar corretamente o contexto em tempo de execução da aplicação e, portanto, escolher adequadamente o segmento a ser baixado, desempenha um papel crítico no gerenciamento de Qualidade de Experiência (QoE) pelo *player* DASH. Tal tarefa é de responsabilidade de um algoritmo ABR. A Tabela 8 descreve os algoritmos utilizados nos experimentos realizados.

Tabela 8: Algoritmos ABR/DASH utilizados nos experimentos.

Algoritmo	**Descrição**
SARA	*Segment Aware Rate Adaptation Algorithm* (SARA) [63] é um algoritmo que considera a ocupação de *buffer*, taxa de transferência média e o tamanho do segmento como métricas para estimar a melhor taxa de bits para os próximos segmentos a serem requisitados. Para requisitar o próximo segmento, SARA estima o tempo de download $T = (s_{curr+1})/H_n$, onde s_{curr+1} é o tamanho em bytes do próximo segmento, e H_n é um média harmônica do tempo de transmissão, dado por $H_n = (\sum_{i=1}^{n} w_i)/\left(\sum_{i=1}^{n} \frac{w_i}{d_i}\right)$, onde o w_i são os pesos proporcionais ao tamanho do segmento e d_i é a taxa de download para o respectivo segmento. Além disso, três limites I, B_α, B_β são associados à ocupação atual do *buffer* para realizar trocas entre bitrates quando necessário. Se $T > B_{curr} - I$ a taxa de bits atual é reduzida, e se $I < B_{curr} \leqslant B_\alpha$ a taxa de bits atual é aumentada.
TBA	*Throughput based Algorihtm* (TBA) [79] é um algoritmo básico, que seleciona o próximo nível de representação do segmento de acordo com a taxa de transmissão experimentada durante a conexão. Ao receber um segmento, o próximo segmento pode ser mudado para uma taxa de bits mais alta, se $\mu > 1 + \epsilon$, onde μ é o parâmetro sensível ao congestionamento (relação entre a duração do segmento e seu tempo de busca), enquanto $i + \epsilon$ é o fator de mudança. Em tal fator, $\epsilon = \max\{(b_{i+1} - b_i)/b_i\}$, onde b_i é a taxa de bits do segmento i. A taxa de bits é reduzida, se $\mu < \lambda_d$, onde λ_d é um limiar de redução.
BBA	*Buffer-based Approach* (BBA) [52] é um algoritmo que seleciona a taxa de bits do próximo segmento (R_{next}) com base na ocupação atual do *buffer* (B_{curr}), medido em tempo de video armazenado. O *buffer* é dividido em duas regiões: reserva (r) e expansão (c), onde $r < c$. Em [52], assume-se valores em torno de 37% e 90% do capacidade do *buffer*, respectivamente, e.g, um *buffer* para 240 s de video, as regiões de r e c poderiam ser de 90 s e 216 s, respectivamente. Se $B_{curr} \leqslant r$, então R_{next} será a taxa de bits mais baixa existente (R_{min}). Se $B_{curr} \geqslant (r + c)$, então R_{next} será a taxa mais alta (R_{max}). Se $r < B_{curr} < c$, uma função $f(.)$ irá maximizar ou minimizar taxas R_i próximas aos limiares de R_- e R_+, onde $r \leqslant R_-$ e $R_+ \leqslant c$. Os limiares são ajustados conforme B_{curr}.

5.3 CENÁRIOS EMULADOS DE GARGALOS DE REDE

Para realizar os experimentos, foram implementados dois cenários no emulador `mininet`[3]:

- **SB** (*Shared Bottleneck*): onde apenas um gargalo é compartilhado entre subfluxos MPQUIC e demais tráfegos de fundo.
- **NSB** (*Non-Shared Bottleneck*): onde os gargalos são disjuntos, não sendo compartilhados entre os subfluxos MPQUIC, mas sendo compartilhados com tráfegos de fundo.

As Figuras 16 e 17 ilustram as topologias dos cenários SB e NSB, respectivamente. As subseções a seguir descrevem detalhes as configurações dos gargalos e a geração de tráfego de fundo (*background traffic*).

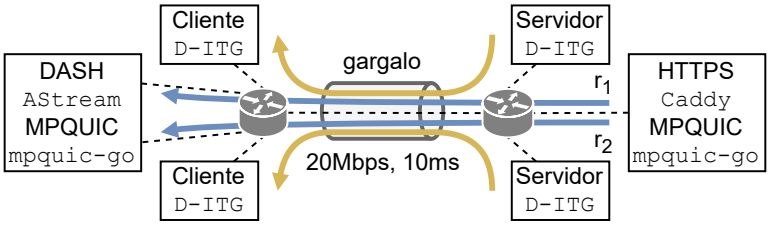


Figura 16: Cenário de gargalos compartilhados SB (*Shared Bottleneck*).

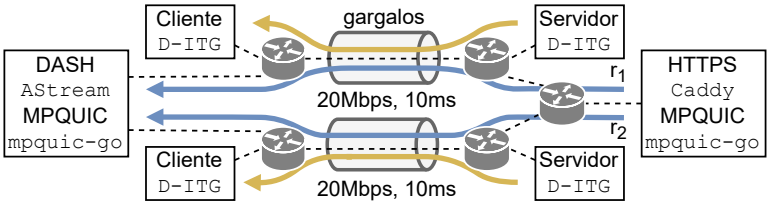


Figura 17: Cenário de gargalos não compartilhados (*Non-Shared Bottleneck*).

5.3.1 *Configuração dos Gargalos*

Os gargalos possuem configurações homogêneas, com capacidade de 20 Mbps e latência de 10 ms, portanto, RTT aproximados a 20 ms. A escolha de uma configuração homogênea é interessante, pois impõe uma maior dificuldade para o algoritmo de agrupamento SBD. Desse modo, os subfluxos MPQUIC são distinguidos pelo algoritmo de agrupamento mediante ao impacto do tráfego dos gargalos nas métricas SBD, e não pelo impacto direto de uma configuração heterogênea predefinida. Outros trabalhos apresentam configurações similares de gargalos em ambientes experimentais [31, 47, 127].

Para realizar as configurações dos gargalos foi utilizada a ferramenta `tc` (*Traffic Control*), disponível em sistemas operacionais Linux. Especificamente, foi aplicado um filtro TBF (*Token Bucket Filter*) junto ao `tc`, de modo a criar enlaces de gargalos

3 http://mininet.org/

que operam em FIFO sob rajadas de 2560 bytes em períodos de 1 ms. O filtro TBF limita o tamanho fila conforme o BPD (*Bandwidth Delay Product*), portanto, a fila é dimensionada para atender o tráfego de pacotes sob taxas de 20 Mbps e latência de 10 ms.

5.3.2 *Geração de Tráfego de Fundo*

Além da aplicação de vídeo, há também dois clientes e dois servidores responsáveis por gerar tráfego de fundo, como ilustram as Figuras 16 e 17. Este tráfego é essencial para tornar os cenários emulados mais realísticos, considerando situações possíveis de compartilhamento de recursos de rede na Internet. Para tanto, o tráfego de fundo é gerado de forma sintética através de

- **Transmissão de Taxa Limitada (TL)**. Transmissões UDP e conexões TCP geradas pela ferramenta `D-ITG` (*Distributed Internet Traffic Generator*) [9]. O tráfego UDP é composto por fluxos de taxas médias distribuídas exponencialmente entre 50 e 150 datagramas/segundo, em intervalos *on*/*off* entre 1 e 5 segundos gerados em distribuição de pareto e exponencial, respectivamente. O tráfego TCP é gerado com taxas contantes de 1000 e 2500 pacotes/segundo. Os tamanhos dos pacotes variaram em torno de uma média de 1000 ± 200 bytes.

- **Transmissão Gulosa (TG)**. Uma aplicação cliente-servidor de conexão TCP permanente, gerando carga ininterrupta na aplicação.

Nesta parametrização para dimensionar o tráfego de fundo, foram utilizadas proporções semelhantes ao tráfego configurado em experimentos de [31].

5.4 ACURÁCIA DO ALGORITMO DE AGRUPAMENTO SBD

Para avaliar acurácia do agrupamento do algoritmo SBD em diferente configurações de tráfego de fundo, foram realizados nove experimentos sob o cenário NSB (Figura 17). A escolha de focar a avaliação de acurácia no cenário de gargalos não compartilhados (NSB) se deu pelo fato de tal cenário impor maiores dificuldades para a correta classificação do algoritmo de agrupamento, quando comparado ao cenário SB. Em NSB, os gargalos, embora disjuntos, possuem as mesmas configurações de capacidade e latência, exigindo que o algoritmo de agrupamento distingua os grupos pelo impacto do tráfego sobre as métricas SBD observadas em cada subfluxo MPQUIC. Tal dificuldade de classificação não foi observada de forma predominante em cenários SB. Em SB, tanto os subfluxos quanto o tráfego de fundo percorrem um gargalo comum, de modo que as métricas SBD observadas nos subfluxos tendem a apresentar comportamento similar, fazendo com que o algoritmo de agrupamento classifique os subfluxos como pertencente ao mesmo grupo – experimentos mostraram taxas de acerto superiores a 98% independente do tráfego de fundo.

Para cada experimento no cenário NSB, foi obtido o percentual de acerto, determinado por

$$acc = \frac{N_{acertos}}{N_{total}} \times 100, \tag{17}$$

onde $N_{acertos}$ é o número de vezes que o algoritmo SBD identificou corretamente que os subfluxos não estão compartilhando gargalo, e o N_{total} é o número de vezes que o algoritmo SBD realizou tentativas de classificação.

5.4.1 *Planejamento dos Experimentos*

O planejamento dos experimentos considerou diferentes combinações de tráfego de fundo, como descreve a Tabela 9. Possíveis combinações foram escolhidas a partir da quantidade de fluxos de taxa limitada (TL) com UDP e TCP e fluxos de transmissões gulosas (LG) com TCP entre os nós clientes e servidores de tráfego de fundo.

Tabela 9: Planejamento dos experimentos de acordo com diferentes configurações de tráfego de fundo (em números de fluxos).

	Gargalo 1			**Gargalo 2**		
Experimento	**UDP–TL**	**TCP–TL**	**TCP–TG**	**UDP–TL**	**TCP–TL**	**TCP–TG**
(1)	4	0	0	4	2	1
(2)	4	1	0	4	2	1
(3)	4	1	0	4	2	0
(4)	4	1	1	4	2	0
(5)	4	0	1	4	2	0
(6)	4	0	1	4	2	1
(7)	4	0	1	4	0	1
(8)	4	1	1	4	2	1
(9)	4	0	1	4	1	1

Legenda: TL (Taxa Limitada), TL (Transmissão Gulosa)

Cada um dos experimentos foi repetido quatro vezes. O tráfego de fundo foi composto por uma base de 4 fluxos UDP de taxa limitada (UDP-TL), sendo acrescida de diferentes quantidades de fluxos TCP de taxa fixa (TCP-TL) e transmissão gulosa (TCP-TG).

5.4.2 *Acurácia Observada*

Os resultados de acurácia são apresentados no gráfico da Figura 18. Como os gargalos possuem configurações idênticas, a maior acurácia na distinção dos subfluxos MPQUIC pelo algoritmo SBD se deu quando o tráfego TCP mais se diferenciou sobre os gargalos, como pode ser observado nos experimentos (1), (5) e (6), com taxas médias de acerto acima de 80%. Com uma maior discrepância do tráfego de fundo, diferentes valores de OWD entre os gargalos foram observados na maior parte do tempo de transmissão, possibilitando ao algoritmo SBD identificar de forma correta que os dois subfluxos MPQUIC estavam fluindo por dois gargalos diferentes.

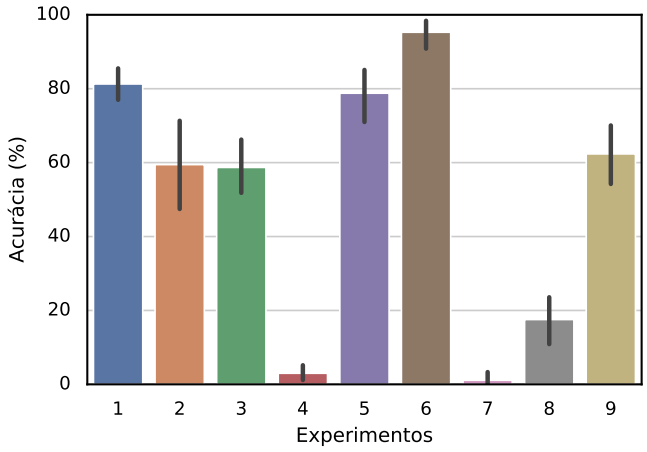


Figura 18: Acurácia do algoritmo de agrupamento SBD em cenário de gargalos não compartilhados (NSB).

Os experimentos (2) e (3) geraram taxas de acerto similares, com médias entre 50% e 70%. As piores taxas foram observadas nos experimentos (4), (7) e (8). Entende-se que o resultado ruim de acurácia foi devido à similaridade do tráfego observado nos dois gargalos, gerando latências de OWD e métricas SBD pouco discrepantes entre os subfluxos MPQUIC e, com isso, uma baixa taxa de acerto do algoritmo SBD para distinguir se o subfluxo está ou não compartilhando gargalo. Os tráfegos dos gargalos se assemelham quando há fluxos TCP guloso e TCP de taxa fixa em ambos gargalos, como nos experimentos (7) e (8). A semelhança de tráfego também ocorre no experimento (4), com dois fluxos TCP em cada gargalo. Embora cenários de gargalos de configuração homogênea e tráfego similar sejam pouco frequentes na Internet, esses cenários são importantes para testar a robustez do algoritmo de agrupamento SBD.

Para seguir com os experimentos de comparação entre as diferentes combinações de soluções MPQUIC e DASH, foi necessário selecionar uma configuração adequada de tráfego de fundo. Dois critérios de seleção foram adotados. O primeiro, os resultados de acurácia deveriam ser compartíveis com resultados observados em trabalhos anteriores [31]. O segundo critério, considerando possíveis situações mais comuns de tráfego na Internet, o tráfego de fundo deveria ser composto em ambos os gargalos por fluxos UDP e TCP guloso. Assim, foi selecionada a configuração de tráfego de fundo do experimento (9), onde cada gargalo assimila 4 fluxos UDP-TL e 1 fluxo TCP-TG, contudo, se diferenciando apenas por 1 fluxo TCP de taxa fixa. No experimento (9), foram observadas taxas de acerto de 60% a 70%, similares as observadas em [31]. Essas taxas são razoáveis, estando fora de cenários e situações ideias (i.e., quando há tráfego bem discrepante entre os gargalos) que, consequentemente, teriam taxas de acerto bem melhores do algoritmo de agrupamento, como nos experimentos (1) e (6).

5.5 QUALIDADE DE SERVIÇO NO TRANSPORTE MPQUIC

Uma vez selecionada a configuração de tráfego de fundo, (9), novos experimentos foram realizados para avaliar o desempenho do MPQUIC-SBD. Tal desempenho foi

medido em termos de vazão e latência observadas nas transmissões dos segmentos de video.

Para comparar MPQUIC-SBD, foram avaliadas duas soluções de *baselines*: LowRTT e NineTails, descritos na Tabela 7. Cada experimento continha uma combinação entre cenário (SB ou NSB), solução MPQUIC (NineTails, LowRTT, SBD), algoritmo ABR/-DASH (SARA, TBA, BBA). Cada experimento foi repetido quatro vezes, totalizando 72 execuções de experimentos.

5.5.1 *Vazão na transmissão do vídeo*

A vazão da transmissão dos segmentos de vídeo observada na aplicação é

$$T_{avg} = \frac{1}{k} \sum_{i}^{k} \frac{s_i}{d_i}, \tag{18}$$

onde s_i é o tamanho do segmento i e d_i é o tempo de *download* desse segmento. A vazão da aplicação corresponde, portanto, à vazão provida pela agregação das transmissões dos subfluxos MPQUIC. A Figura 19 apresenta o gráfico das médias de vazões observadas nas transmissões de segmentos de video.

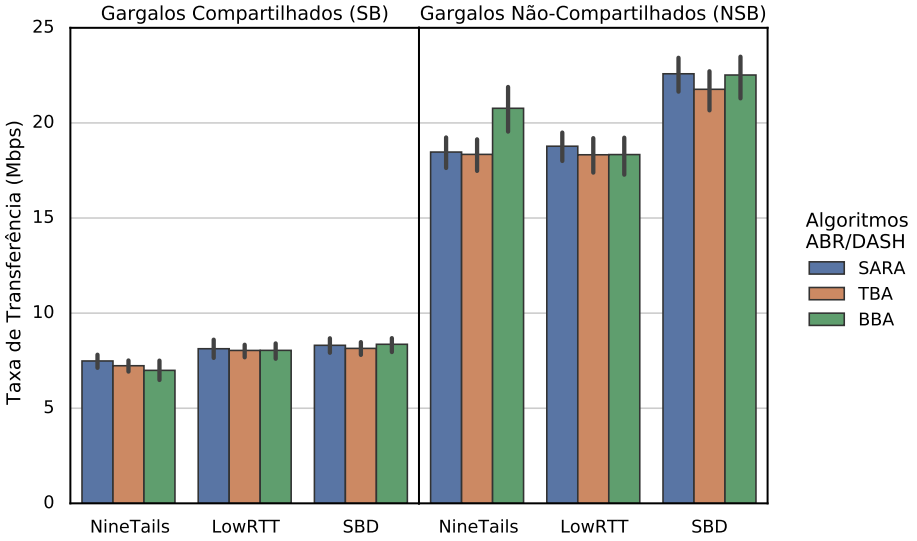


Figura 19: Vazão da transmissão dos segmentos de vídeo.

No cenário de gargalos compartilhados (SB), os desempenhos de vazão e latência obtidos com SBD foram bastantes similares aos obtidos com solução padrão LowRTT. Este resultado era esperado, uma vez que no cenário SB a taxa de acerto do algoritmo de agrupamento SBD é bastante elevada ($> 98\%$), o que faz com que o MPQUIC mantenha sua operação padrão, ou seja, o SBD não desacopla os subfluxos no controle de congestionamento OLIA. Já o escalonador NineTails obteve vazão inferior ao LowRTT e SBD no cenário SB, o que também era esperado. Tal escalonador é redundante ao replicar os quadros do fim das mensagens sobre os subfluxos MPQUIC existentes.

No cenário de gargalos não compartilhados (NSB), o mecanismo SBD possibilitou maior vazão na transmissão de segmentos de vídeo, em torno 23 Mbps comparado

aos 17 Mbps observados sob LowRTT e NineTails. Esse melhor desempenho, com ganhos em torno de 25% de vazão, é decorrente da operação desacoplada dos subfluxos MPQUIC no controle de congestionamento OLIA, visto que SBD permite detectar corretamente que os subfluxos não pertencem ao mesmo grupo, portanto, não compartilham o mesmo gargalo. Observa-se que o BBA em NineTails se destacou em relação ao TBA e SARA. O escalonador NineTails possibilita menor latência, e às vezes ganho na vazão, pois a replicação de pacotes sobre os subfluxos ocorre no final das mensagens, havendo agregação como em LowRTT antes disso. Um menor tempo de download dos segmentos em BBA/Ninetails no cenário NSB é observado na Figura 20. Contudo, a qualidade média do vídeo requisitado com BBA/Ninetails é inferior ao TBA e SARA, como pode ser visto na Figura 21, de modo que segmentos que foram requisitados, ainda que menores, puderam atingir maior vazão.

5.5.2 *Latência da transmissão do vídeo*

A latência média da transmissão dos segmentos de vídeo é

$$D_{avg} = \frac{1}{k} \sum_{i}^{k} d_i, \tag{19}$$

onde d_i é o tempo de *download* de um segmento de video i. A Figura 20 apresenta o gráfico das médias das latências observadas nas transmissões de segmentos de video.

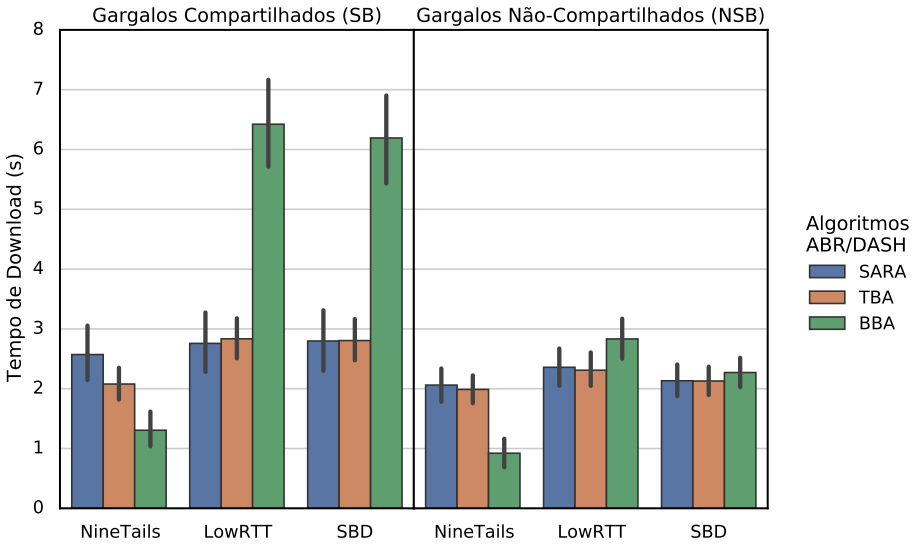


Figura 20: Latência de transmissão dos segmentos de vídeo.

Embora a transmissão redundante em NineTails não se beneficie da agregação dos subfluxos, ela possibilita uma menor latência de transmissão, como mostra os gráficos na Figura 20, já que o destinatário sempre recebe quadros duplicados, armazenando o primeiro quadro que chega pelo subfluxo mais rápido e descartando os demais. No cenário de gargalo compartilhado (SB), o algoritmo BBA resultou em transmissões de maiores latências para os segmentos de vídeo em ambas soluções LowRTT e SBD. Isso ocorreu, pois, o algoritmo BBA requisitou segmentos de vídeo de maior

qualidade (*bitrate*), portanto, segmentos de vídeos maiores que demandaram tempos maiores de *download*.

5.6 QUALIDADE DE EXPERIÊNCIA NA APLICAÇÃO DE VIDEO

Um segmento de video tem um tamanho em bytes, que representa uma compressão em taxa constante de bits por segundo (*bitrate*), de modo que quanto maior o segmento (em bytes), maior é o seu *bitrate*. A aplicação de *player* DASH, sob algum critério bem definido, irá variar a qualidade do vídeo, ou seja, irá requisitar segmentos de video de diferentes taxas de bits ao servidor durante a reprodução (*streaming*) do video. A Tabela 8 descreve os algoritmos de ABR (*Adaptive Bit Rate*) mais populares em aplicações DASH, os quais utilizamos nos experimentos.

5.6.1 *Taxas de bits dos segmentos requisitados*

A qualidade do vídeo, portanto, é crucial para uma boa experiência para o usuário. O ponto é o que define a qualidade do vídeo. Uma métrica é a média observada do *bitrate* dos segmentos requisitados ao longo do *streaming* de vídeo, ou a qualidade média dos segmentos [53]:

$$Q_{avg} = \frac{1}{k} \sum_{i=1}^{k} Q_i, \tag{20}$$

onde Q_i é o *bitrate* do segmento i requisitado, sendo k o número total de segmentos requisitados durante a reprodução do vídeo.

A Figura 21 apresenta o gráfico da qualidade média dos segmentos requisitados pela aplicação DASH. Considerando o impacto das soluções de transporte MPQUIC na qualidade média dos segmentos, observa-se que o SBD possibilitou melhorias evidentes no cenário de gargalo não-compartilhado (NSB) para os algoritmos TBA e SARA. Esses algoritmos levam em consideração as condições da rede, de modo que comportamento da qualidade do segmento é proporcional à vazão observada no MPQUIC (Figura 19), i.e., quanto maior a vazão de transmissão maior o *bitrate* segmento requisitado. No caso do algoritmo SARA, não apenas considera a taxa de transmissão observada na rede, como o TBA, mas também a ocupação do *buffer* e o tamanho do segmento.

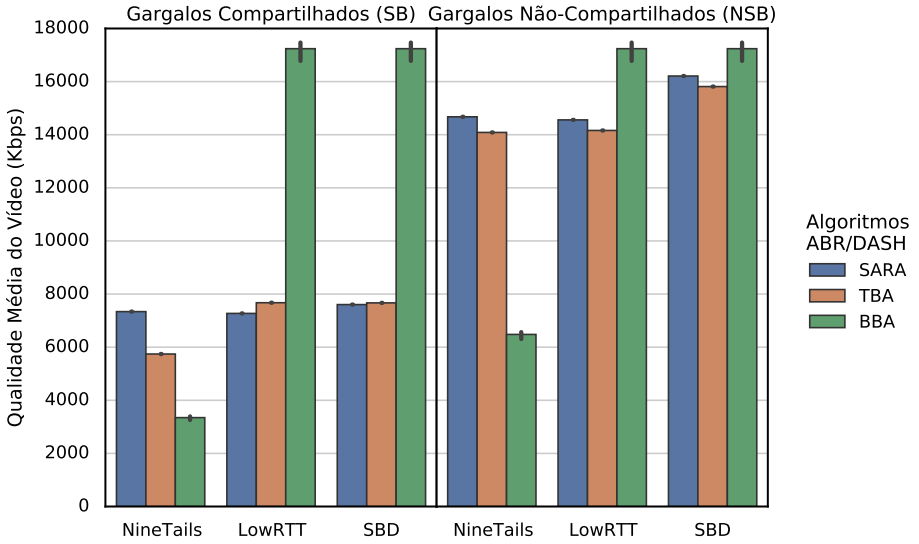


Figura 21: Qualidade média (*bitrate*) dos segmentos de vídeo requisitados.

Por outro lado, o *bitrate* de segmentos requisitados com BBA foram maiores, independente dos cenários (SB ou NSB) e das soluções de transporte MPQUIC. Particularmente, o algoritmo BBA busca o próximo segmento de video baseado na atual ocupação do *buffer*, sendo mensurada em segundos de video já armazenados. Especificamente, o *bitrate* aumenta em função da ocupação do *buffer*, i.e., quanto mais tempo de vídeo estiver armazenado no *buffer*, mais seguro é requisitar segmentos *bitrates* maiores.

5.6.2 *Frequência de troca de taxas de bits*

Outra medida de QoE é a frequência de troca de qualidade de segmentos, dada por [53]:

$$F_{avg} = \frac{1}{k} \sum_{i=1}^{k} F(Q_i), \tag{21}$$

onde

$$F(Q_i) = \begin{cases} 1, & \text{se } i = 1, \\ 1, & \text{se } Q_{i-1} \neq Q_i, \\ 0, & \text{caso contrário.} \end{cases} \tag{22}$$

A frequência $F(Q_i)$ assume valor 1 se o *bitrate* dos segmentos anterior e atual foram diferentes, e valor de 0 se eles forem iguais. A frequência média de troca F_{avg} é calculada sobre os k segmentos requisitados durante a reprodução do vídeo, resultando em valores no intervalo de zero e um.

A Figura 22 apresenta o gráfico com resultados de F_{avg}. No cenário de gargalos compartilhados (SB), observa-se que o algoritmo BBA, ao requisitar segmentos de maiores *bitrate* (Figura 21), possibilitou menor frequência de trocas de qualidade de segmentos durante a reprodução do vídeo. As trocas foram inferiores a 20% das requisições de video para BBA, em torno 70% para SARA e acima de 90% para TBA.

Tanto o SARA quanto TBA levam em consideração a latência e vazão de transmissão, respectivamente, dos segmentos requisitados. Dessa forma, uma alta frequência de troca nesses algoritmos reflete maior oscilação da condição da rede.

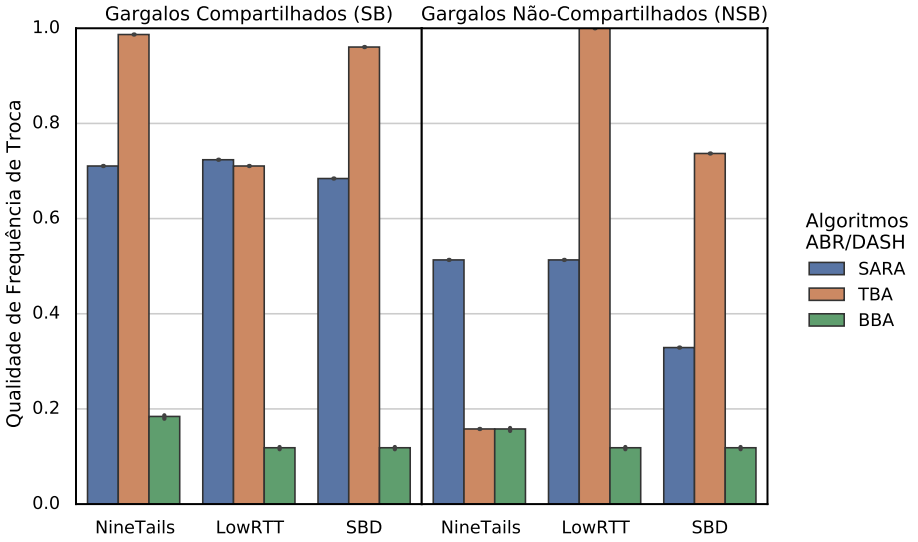


Figura 22: Frequência de troca de qualidade de segmentos.

No cenário de gargalos não compartilhados (NSB), houve maior redução da frequência de troca em transmissões do SBD, principalmente para os algoritmos SARA em TBA, em comparação com as soluções MPQUIC de NineTails e LowRTT.

Na interpretação dos resultados com o indicador F_{avg} deve-se considerar que, por mais que a frequência de troca seja menor, não quer dizer que realmente possibilitou uma melhor qualidade de experiência para o usuário. Por exemplo, no transporte de NineTails, o algoritmo BBA requisitou mais segmentos de *bitrates* menores (Figura 21), observando também frequências menores de trocas de qualidade. Além disso, não seria ruim para o usuário quando o algoritmo ABR requisita um segmento de vídeo de maior qualidade que o anterior. Essa situação não é considerada pelo indicador F_{avg}. Nesse contexto, a análise do indicador F_{avg} é requer ser acompanhada da análise de outros indicadores para possibilitar melhor compreensão da qualidade de experiência observada na transmissão de vídeo.

5.6.3 *Rebufferização durante a reprodução do vídeo*

Rebufferização são as interrupções observadas durante a reprodução do vídeo. Essas interrupções, em geral, possuem razões associadas à operação dos algoritmos ABR e às condições da rede. Por exemplo, se o algoritmo ABR se mantém requisitando segmentos muito grandes para a rede sustentar (i.e., segmentos de video de *bitrate* muito elevados), em algum momento o *player* irá consumir todo o *buffer* e a reprodução do video congela, exibindo a mensagem de "Rebuffering ..." na tela [52].

Durante sua execução da aplicação de vídeo, a exibição do vídeo pode estar em dois possíveis estados: *tocando* (*playback*), ou interrompido (*rebuffering*). Dessa forma, a frequência de rebufferização observada durante a reprodução do vídeo é dada por:

$$F_{rebuf} = \frac{N_{rebuf}}{N_{rebuf} + N_{play}}, \tag{23}$$

onde N_{rebuf} é total de estados de interrupções e N_{play} é o total estados em *playback* registrados durante a reprodução do vídeo. A Figura 23 apresenta a média das frequências de rebufferização observada.

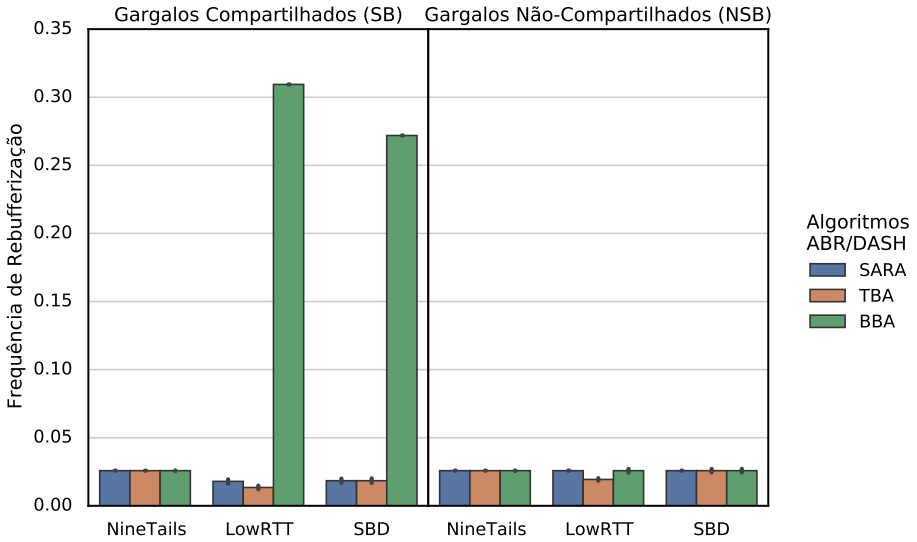


Figura 23: Frequência de rebufferização durante a reprodução do vídeo.

O algoritmo BBA possibilitou maiores *bitrates* (Figure 21) no geral, contudo, ao custo de frequências elevadas, entre 27% e 31%, de rebuferização no cenário de gargalos compartilhados. No cenário SB, os segmentos de vídeo requisitados foram grandes para as condições da rede observadas com o transporte MPQUIC de ambos LowRTT e SBD. Embora NineTails tenha provido baixa frequência de rebufferização, o *bitrate* dos segmentos requisitados foi baixo (Figure 21). No cenário de gargalos não compartilhados (NSB), em geral, a frequência de rebufferização foi baixa, inferior a 3%.

5.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou discussão de resultados experimentais conduzidos na avaliação do mecanismo SBD implementado. Os experimentos consideraram transmissões de vídeo, que são transmissões bastante comuns atualmente, principalmente, por usuários de dispositivos portáteis, mas pouco conhecidas junto ao protocolo MPQUIC.

Em cenários de gargalos compartilhados, a elevada acurácia ($> 98\%$) do algoritmo de agrupamento SBD fez com que os subfluxos fossem normalmente acoplados no controle de congestionamento OLIA, mantendo assim o compartilhamento justo dos recursos de redes. Nos cenários de gargalos não compartilhados, o mecanismo SBD

possibilitou melhor desempenho na transmissão de segmentos de vídeos, desacoplando corretamente os subfluxos no controle de congestionamento *multipath*. Particularmente, considerando configurações de tráfego de fundo escolhido, mesmo com acurácia mais modestas ($\approx$ 60%) na classificação de agrupamento em cenários de gargalos não compartilhados, o mecanismo SBD possibilitou ganhos de 25% na vazão da transmissão dos segmentos de vídeo quando comparado a outras soluções de transporte MPQUIC, como escalonadores padrão LowRTT e NineTails. A melhor qualidade de serviço na transmissão possibilitou melhores indicadores de qualidade de experiência em diferentes algoritmos ABR na aplicação de vídeo.

6 CONCLUSÕES

Neste trabalho foi realizada uma investigação sobre a detecção de compartilhamento de gargalos junto ao protocolo MPQUIC [20]. Uma extensão do protocolo MPQUIC em linguagem Go [21], denominada MPQUIC-SBD, foi implementada e validada para prover o suporte de um mecanismo SBD conforme padrão de Internet especificado na RFC 8382 [46]. Enquanto a detecção de compartilhamento de gargalos possui um estado maduro nos protocolos TCP e MPTCP, pouco se conhece dos seus efeitos em transmissões do protocolo MPQUIC, sobretudo, em transmissões de aplicações de vídeo como foram experimentalmente avaliadas. Resultados experimentais a partir de emulações de transmissões DASH mostraram ganhos de desempenho em QoS que se refletem em melhor QoE para as aplicações de vídeo.

A implementação do mecanismo MPQUIC-SBD está disponível em [94], possibilitando que o mecanismo possa ser investigado e aprimorado por outros pesquisadores. Um artigo científico [93] foi publicado no Workshop WPIETF 2020 (VII Workshop Pré-IETF), recebendo menção dentre os melhores artigos do evento. No momento, um novo artigo está sendo escrito, considerando a discussão dos resultados experimentais de transmissões de vídeo apresentados no Capítulo 5.

Como trabalhos futuros, a investigação iniciada aqui tem outros desdobramentos previstos. Um deles é a avaliação de classificadores de subfluxos, utilizando as métricas SBD como entrada para algoritmos e heurísticas baseadas em aprendizado de máquina. Outra investigação futura será na avaliação da robustez de MPQUIC-SBD, considerando outros fluxos de transmissão, além de segmento de vídeo, como fluxos elefantes (e.g., grandes arquivos) e fluxos ratos (e.g., navegação web).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ali, A. A.; Michaut, F.; Lepage, F. End-to-end available bandwidth measurement tools : A comparative evaluation of performances. *CoRR*, v. abs/0706.4004, 2007.
Disponível em http://arxiv.org/abs/0706.4004 (Acessado em 5 jun. 2020)

[2] Allman, M. Measuring end-to-end bulk transfer capacity. In: *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, New York, NY, USA: ACM, 2001, p. 139–143 (*IMW '01*, v.).
Disponível em http://doi.acm.org/10.1145/505202.505220 (Acessado em 5 jun. 2020)

[3] Allman, M.; Paxson, V.; Blanton, E. *TCP Congestion Control*. Internet Standart RFC 5681, IETF, 2009.

[4] Almes, G.; Kalidindi, S.; Zekauskas, M. *A one-way packet loss metric for ippm*. RFC 2680, RFC Editor, 1999.

[5] Alvestrand, H. T. *Overview: Real Time Protocols for Browser-based Applications*. Internet-Draft draft-ietf-rtcweb-overview-19, Internet Engineering Task Force, work in Progress, 2017.
Disponível em https://datatracker.ietf.org/doc/html/draft-ietf-rtcweb-overview-19 (Acessado em 5 jun. 2020)

[6] Amante, S.; Carpenter, B.; Jiang, S.; Rajahalme, J. *Ipv6 flow label specification*. RFC 6437, RFC Editor, 2011.

[7] Bellovin, S. M. A best-case network performance model. *ATT Research, Tech. Rep*, 1992.

[8] Bolot, J.-C. Characterizing end-to-end packet delay and loss in the internet. *J. High Speed Netw.*, v. 2, n. 3, p. 305–323, 1993.
Disponível em http://dl.acm.org/citation.cfm?id=2692175.2692182 (Acessado em 5 jun. 2020)

[9] Botta, A.; Dainotti, A.; Pescapé, A. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, v. 56, n. 15, p. 3531–3547, 2012.
Disponível em https://www.sciencedirect.com/science/article/pii/S1389128612000928 (Acessado em 5 April. 2021)

[10] Brownlee, N. *Srl: A language for describing traffic flows and specifying actions for flow groups*. RFC 2723, RFC Editor, 1999.

[11] Brownlee, N.; Mills, C.; Ruth, G. *Traffic flow measurement: Architecture*. RFC 2063, RFC Editor, 1997.

[12] Carter, R. L.; Crovella, M. E. *Dynamic server selection using bandwidth probing in wide-area networks*. Relatório Técnico, Citeseer, 1996.

[13] Carter, R. L.; Crovella, M. E. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, v. 27-28, p. 297 – 318, 1996. Disponível em http://www.sciencedirect.com/science/article/pii/S0166531696900322 (Acessado em 5 jun. 2020)

[14] Chaudet, C.; Fleury, E.; Lassous, I. G.; Rivano, H.; Voge, M.-E. Optimal positioning of active and passive monitoring devices. In: *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, CoNEXT '05, New York, NY, USA: ACM, 2005, p. 71–82 (*CoNEXT '05*, v.). Disponível em http://doi.acm.org/10.1145/1095921.1095932 (Acessado em 5 jun. 2020)

[15] Chimento, P.; Ishac, J. *Defining network capacity*. RFC 5136, RFC Editor, 2008.

[16] Cisco Cisco visual networking index: forecast and methodology, 2017-2022. 2017.

[17] Claise, B. Cisco Systems NetFlow Services Export Version 9. RFC 3954, 2004. Disponível em https://rfc-editor.org/rfc/rfc3954.txt (Acessado em 5 jun. 2020)

[18] Claise, B. *Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information*. RFC 5101, RFC Editor, http://www.rfc-editor.org/rfc/rfc5101.txt, 2008. Disponível em http://www.rfc-editor.org/rfc/rfc5101.txt (Acessado em 5 jun. 2020)

[19] Coates, A.; Hero III, A. O.; Nowak, R.; Bin Yu Internet tomography. *IEEE Signal Processing Magazine*, v. 19, n. 3, p. 47–65, 2002.

[20] Coninck, Q. D.; Bonaventure, O. *Multipath Extensions for QUIC (MP-QUIC)*. Internet-Draft draft-deconinck-quic-multipath-04, Internet Engineering Task Force, work in Progress, 2020. Disponível em https://datatracker.ietf.org/doc/html/draft-deconinck-quic-multipath-04 (Acessado em 5 jun. 2020)

[21] De Coninck, Q.; Bonaventure, O. Multipath quic: Design and evaluation. In: *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, New York, NY, USA: Association for Computing Machinery, 2017, p. 160–166 (*CoNEXT '17*, v.). Disponível em https://doi.org/10.1145/3143361.3143370 (Acessado em 5 jun. 2020)

[22] Dierks, T.; Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, 2006. Disponível em https://rfc-editor.org/rfc/rfc4346.txt (Acessado em 5 jun. 2020)

[23] Dovrolis, C.; Prasad, R. Pathrate: A measurement tool for the capacity of network paths. *URL: www. cc. gatech. edu/fac/Constantinos. Dovrolis*, 2004.

[24] Dovrolis, C.; Ramanathan, P.; Moore, D. What do packet dispersion techniques measure? In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, 2001, p. 905–914 vol.2.

[25] Downey, A. B. Clink: a tool for estimating internet link characteristics. 1999.

[26] Downey, A. B. Using pathchar to estimate internet link characteristics. In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '99, New York, NY, USA: ACM, 1999, p. 241–250 (*SIGCOMM '99*, v.).
Disponível em http://doi.acm.org/10.1145/316188.316228 (Acessado em 5 jun. 2020)

[27] En-Najjary, T.; Urvoy-Keller, G. Pprate: A passive capacity estimation tool. In: *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, 2006, p. 82–89.

[28] Ericsson Streaming video - from megabits to gigabytes. https://www.ericsson.com/4a30ac/assets/local/mobility-report/documents/2018/emr-nov-2018-streaming-video.pdf, accessed: 2021-04-7, 2018.

[29] Ervasti, K. A survey on network measurement: Concepts, techniques, and tools. *University of Helsinki*, 2016.

[30] Ferlin, S.; Alay, Ö.; Dreibholz, T.; Hayes, D. A.; Welzl, M. Revisiting congestion control for multipath tcp with shared bottleneck detection. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, p. 1–9.

[31] Ferlin, S.; Alay, Ö.; Dreibholz, T.; Hayes, D. A.; Welzl, M. Revisiting congestion control for multipath TCP with shared bottleneck detection. In: *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016)*, 2016, p. 1–9.

[32] Floyd, S. *Metrics for the evaluation of congestion control mechanisms*. RFC 5166, RFC Editor, http://www.rfc-editor.org/rfc/rfc5166.txt, 2008.
Disponível em http://www.rfc-editor.org/rfc/rfc5166.txt (Acessado em 5 jun. 2020)

[33] F.M., P.; K., S.; B.M., T.; Z, S. Improved models of internet charging scheme of single bottleneck link in multi qos networks. *Journal of Applied Sciences*, 2013.
Disponível em http://ddms.usim.edu.my:80/jspui/handle/123456789/15429 (Acessado em 5 jun. 2020)

[34] Ford, A.; Raiciu, C.; Handley, M.; Barre, S.; Iyengar, J. *Architectural Guidelines for Multipath TCP Development*. Internet Standart RFC 6182, IETF, 2011.

[35] FORD, A.; RAICIU, C.; HANDLEY, M. J.; BONAVENTURE, O. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, 2013.
Disponível em https://rfc-editor.org/rfc/rfc6824.txt (Acessado em 5 jun. 2020)

[36] FREIER, A. O.; KARLTON, P.; KOCHER, P. C. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101, 2011.
Disponível em https://rfc-editor.org/rfc/rfc6101.txt (Acessado em 5 jun. 2020)

[37] FUENTES, F.; KAR, D. C. Ethereal vs. tcpdump: A comparative study on packet sniffing tools for educational purpose. *J. Comput. Sci. Coll.*, v. 20, n. 4, p. 169–176, 2005.
Disponível em http://dl.acm.org/citation.cfm?id=1047846.1047873 (Acessado em 5 jun. 2020)

[38] GOYAL, P.; GOYAL, A. Comparative study of two most popular packet sniffing tools-tcpdump and wireshark. In: *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2017, p. 77–81.

[39] GRÖHN, A. *Clock synchronisation of a computer test network (testiverkon kellosynkronointi)*. Master's thesis, Networking Laboratory, Helsinki University of Technology, 2004.
Disponível em http://www.netlab.hut.fi/tutkimus/ironet/papers/diploma_thesis_grohn.pdf (Acessado em 5 jun. 2020)

[40] HA, S.; RHEE, I.; XU, L. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, v. 42, n. 5, p. 64–74, 2008.
Disponível em https://doi.org/10.1145/1400097.1400105 (Acessado em 5 jun. 2020)

[41] HAILI, L.; KE, Z.; WANWEI, H. Key technologies of network traffic measurement. In: *2014 IEEE 5th International Conference on Software Engineering and Service Science*, 2014, p. 1093–1098.

[42] HAITAO WANG; LIHUA SONG; HUI CHEN; LI YAN Study on network measurement technologies and performance evaluation methods. In: *2013 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, 2013, p. 377–380.

[43] HALL, J. *Multi-layer network monitoring and analysis*. Relatório Técnico UCAM-CL-TR-571, University of Cambridge, Computer Laboratory, 2003.
Disponível em https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-571.pdf (Acessado em 5 jun. 2020)

[44] HANDLEY, M.; FLOYD, S.; PADHYE, J.; WIDMER, J. *Tcp friendly rate control (tfrc): Protocol specification*. RFC 3448, RFC Editor, 2003.

[45] HASSAYOUN, S.; IYENGAR, J.; ROS, D. Dynamic window coupling for multipath congestion control. In: *2011 19th IEEE International Conference on Network Protocols*, 2011, p. 341–352.

[46] HAYES, D.; FERLIN, S.; WELZL, M.; HIORTH, K. Shared Bottleneck Detection for Coupled Congestion Control for RTP Media. RFC 8382, 2018. Disponível em https://rfc-editor.org/rfc/rfc8382.txt (Acessado em 5 jun. 2020)

[47] HAYES, D. A.; FERLIN, S.; WELZL, M. Practical passive shared bottleneck detection using shape summary statistics. In: *39th Annual IEEE Conference on Local Computer Networks*, 2014, p. 150–158.

[48] HEDAYAT, K.; KRZANOWSKI, R.; MORTON, A.; YUM, K.; BABIARZ, J. *A two-way active measurement protocol (twamp)*. RFC 5357, RFC Editor, 2008.

[49] HIDESHIMA, Y.; KOIKE, H. Starmine: A visualization system for cyber attacks. In: *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, AUS: Australian Computer Society, Inc., 2006, p. 131–138 (*APVis '06*, v.).

[50] HOFSTEDE, R.; ČELEDA, P.; TRAMMELL, B.; DRAGO, I.; SADRE, R.; SPEROTTO, A.; PRAS, A. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys Tutorials*, v. 16, n. 4, p. 2037–2064, 2014.

[51] HU, N.; LI, L.; MAO, Z. M.; STEENKISTE, P.; WANG, J. A measurement study of internet bottlenecks. In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 2005, p. 1689–1700 vol. 3.

[52] HUANG, T.-Y.; JOHARI, R.; MCKEOWN, N.; TRUNNELL, M.; WATSON, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *SIGCOMM Comput. Commun. Rev.*, v. 44, n. 4, p. 187–198, 2014. Disponível em https://doi.org/10.1145/2740070.2626296 (Acessado em 5 April. 2021)

[53] HWANG, M.; PARK, J.; CHUNG, K. Dynamic block-based http adaptive streaming scheme in multiple wireless networks. *IEEE Access*, v. 8, p. 138447–138455, 2020.

[54] IYENGAR, J.; SWETT, I. *QUIC Loss Detection and Congestion Control*. Internet-Draft draft-ietf-quic-recovery-29, Internet Engineering Task Force, work in Progress, 2020. Disponível em https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-29 (Acessado em 5 jun. 2020)

[55] IYENGAR, J.; THOMSON, M. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Internet-Draft draft-ietf-quic-transport-29, Internet Engineering Task Force, work in Progress, 2020. Disponível em https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-29 (Acessado em 5 jun. 2020)

[56] JACOBSON, V. Congestion avoidance and control. In: *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, New York, NY, USA: ACM, 1988, p. 314–329 (*SIGCOMM '88*, v.).

Disponível em http://doi.acm.org/10.1145/52324.52356 (Acessado em 5 jun. 2020)

[57] Jacobson, V. Congestion avoidance and control. In: *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, New York, NY, USA: Association for Computing Machinery, 1988, p. 314–329 (*SIGCOMM '88*, v.). Disponível em https://doi.org/10.1145/52324.52356 (Acessado em 5 jun. 2020)

[58] Jacobson, V. Pathchar : A tool to infer characteristics of internet paths. In: *Presented at the Mathematical Sciences Research Institute (MSRI*, 1997.

[59] Jacobson, V.; Leres, C.; McCanne, S. The tcpdump manual page. *Lawrence Berkeley Laboratory, Berkeley, CA*, v. 143, 1989.

[60] Jain, M.; Dovrolis, C. Pathload: A measurement tool for end-to-end available bandwidth. In: *In Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002, p. 14–25.

[61] Jain, M.; Dovrolis, C. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, v. 11, n. 4, p. 537–549, 2003.
Disponível em http://dx.doi.org/10.1109/TNET.2003.815304 (Acessado em 5 jun. 2020)

[62] Jones, R.; *et al.* Netperf: a network performance benchmark. *Information Networks Division, Hewlett-Packard Company*, 1996.

[63] Juluri, P.; Tamarapalli, V.; Medhi, D. Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http. In: *2015 IEEE International Conference on Communication Workshop (ICCW)*, 2015, p. 1765–1770.

[64] Juluri, P.; Tamarapalli, V.; Medhi, D. Qoe management in dash systems using the segment aware rate adaptation algorithm. In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, p. 129–136.

[65] Karn, P.; Partridge, C. Improving round-trip time estimates in reliable transport protocols. *ACM Trans. Comput. Syst.*, v. 9, n. 4, p. 364–373, 1991.
Disponível em https://doi.org/10.1145/118544.118549 (Acessado em 5 jun. 2020)

[66] Katabi, D.; Bazzi, I.; Xiaowei Yang A passive approach for detecting shared bottlenecks. In: *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, 2001, p. 174–181.

[67] Keshav, S. A control-theoretic approach to flow control. In: *Proceedings of the Conference on Communications Architecture &Amp; Protocols*, SIGCOMM '91, New York, NY, USA: ACM, 1991, p. 3–15 (*SIGCOMM '91*, v.).
Disponível em http://doi.acm.org/10.1145/115992.115995 (Acessado em 5 jun. 2020)

[68] KHALILI, R.; GAST, N.; POPOVIC, M. *Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP*. Internet Draft draft-khalili-mptcp-congestion-control-05, IETF, 2014.

[69] KHALILI, R.; GAST, N.; POPOVIC, M.; YVES LE BOUDEC, J. Opportunistic linked-increases congestion control algorithm for mptcp. 2013.

[70] KIMURA, B. Y. L.; LIMA, D. C. S. F.; VILLAS, L. A.; LOUREIRO, A. A. F. Inter-path contention in multipath tcp disjoint paths. *IEEE/ACM Transactions on Networking*, v. 27, n. 4, p. 1387–1400, 2019.

[71] KREUTZ, D.; RAMOS, F. M.; VERISSIMO, P. Towards secure and dependable software-defined networks. In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, p. 55–60.

[72] LAI, K.; BAKER, M. Measuring link bandwidths using a deterministic model of packet delay. *SIGCOMM Comput. Commun. Rev.*, v. 30, n. 4, p. 283–294, 2000. Disponível em http://doi.acm.org/10.1145/347057.347557 (Acessado em 5 jun. 2020)

[73] LAI, K.; BAKER, M. Measuring link bandwidths using a deterministic model of packet delay. In: *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, New York, NY, USA: ACM, 2000, p. 283–294 (*SIGCOMM '00*, v.). Disponível em http://doi.acm.org/10.1145/347059.347557 (Acessado em 5 jun. 2020)

[74] LAI, K.; BAKER, M. Nettimer: A tool for measuring bottleneck link, bandwidth. In: *Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, Berkeley, CA, USA: USENIX Association, 2001, p. 11–11 (*USITS'01*, v.). Disponível em http://dl.acm.org/citation.cfm?id=1251440.1251451 (Acessado em 5 jun. 2020)

[75] LAMPING, U.; WARNICKE, E. Wireshark user's guide. *Interface*, v. 4, n. 6, 2004.

[76] LANGLEY, A.; RIDDOCH, A.; WILK, A.; VICENTE, A.; KRASIC, C.; ZHANG, D.; YANG, F.; KOURANOV, F.; SWETT, I.; IYENGAR, J.; BAILEY, J.; DORFMAN, J.; ROSKIND, J.; KULIK, J.; WESTIN, P.; TENNETI, R.; SHADE, R.; HAMILTON, R.; VASILIEV, V.; CHANG, W.-T.; SHI, Z. The quic transport protocol: Design and internet-scale deployment. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, New York, NY, USA: Association for Computing Machinery, 2017, p. 183–196 (*SIGCOMM '17*, v.). Disponível em https://doi.org/10.1145/3098822.3098842 (Acessado em 5 jun. 2020)

[77] LENZ, P.; DA SILVA, M. M.; NOGUEIRA, M.; DOS SANTOS, A. Avaliação do problema de reordenamento de pacotes e das políticas de escalonamento em protocolos multicaminhos. In: *Anais do XXIV Workshop de Gerência e Operação de Redes e Serviços*, Porto Alegre, RS, Brasil: SBC, 2019, p. 1–14. Disponível em https://sol.sbc.org.br/index.php/wgrs/article/view/7679 (Acessado em 5 jun. 2020)

[78] Li, M.; Lukyanenko, A.; Ou, Z.; Ylä-Jääski, A.; Tarkoma, S.; Coudron, M.; Secci, S. Multipath transmission for the internet: A survey. *IEEE Communications Surveys Tutorials*, v. 18, n. 4, p. 2887–2925, 2016.

[79] Liu, C.; Bouazizi, I.; Gabbouj, M. Rate adaptation for adaptive http streaming, p. 169–174. 2011.
Disponível em https://doi.org/10.1145/1943552.1943575 (Acessado em 5 April. 2021)

[80] Lv, C.; Ma, B.; Du, X.; Li, B.; Liang, T. Analysis and research of network measurement technologies. In: *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, 2015, p. 117–121.

[81] Maier, G.; Feldmann, A.; Paxson, V.; Allman, M. On dominant characteristics of residential broadband internet traffic. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, New York, NY, USA: ACM, 2009, p. 90–102 (*IMC '09*, v.).
Disponível em http://doi.acm.org/10.1145/1644893.1644904 (Acessado em 5 jun. 2020)

[82] Mao, Z. M.; So, H.-s. W.; Kang, B. Network support for mobile multimedia using a self-adaptive distributed proxy. In: *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '01, New York, NY, USA: ACM, 2001, p. 107–116 (*NOSSDAV '01*, v.).
Disponível em http://doi.acm.org/10.1145/378344.378360 (Acessado em 5 jun. 2020)

[83] Mar, B. pchar : A tool for measuring internet path characteristics. 2000.

[84] Mathis, M. Diagnosing internet congestion with a transport layer performance tool. In: *Pittsburgh Supercomputing Center*, 1996.

[85] Mathis, M.; Allman, M. *A framework for defining empirical bulk transfer capacity metrics*. RFC 3148, RFC Editor, 2001.

[86] McCanne, S.; Jacobson, V. The bsd packet filter: A new architecture for user-level packet capture. In: *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, USENIX'93, USA: USENIX Association, 1993, p. 2 (*USENIX'93*, v.).

[87] Melander, B.; Bjorkman, M.; Gunningberg, P. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In: *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, 2000, p. 415–420 vol.1.

[88] Mills, D. L. Rfc0958: Network time protocol (ntp). 1985.

[89] Mochalski, K.; Irmscher, K. On the use of passive network measurements for modeling the internet. In: Irmscher, K.; Fähnrich, K.-P., eds. *Kommunikation in Verteilten Systemen (KiVS)*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, p. 371–382.

[90] MOHAN, V.; REDDY, Y. R. J.; KALPANA, K. Active and passive network measurements: A survey. 2012.

[91] NINGNING HU; STEENKISTE, P. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, v. 21, n. 6, p. 879–894, 2003.

[92] PAGANINI, F.; DOYLE, J.; LOW, S. Scalable laws for stable network congestion control. In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, 2001, p. 185–190 vol.1.

[93] PAIVA, T.; FERLIN, S.; KIMURA, B. Métricas de detecção de gargalos compartilhados em transmissões por múltiplos caminho em mpquic. In: *Anais do VII Workshop Pré-IETF*, Porto Alegre, RS, Brasil: SBC, 2020, p. 15–28.
Disponível em https://sol.sbc.org.br/index.php/wpietf/article/view/13793

[94] PAIVA, T.; FERLIN, S.; KIMURA, B. MultiPath QUIC - SBD. https://github.com/thomaswpp/mpquic-sbd, accessed: 2021-04-10, 2021.

[95] PÁSZTOR, A.; VEITCH, D. Active probing using packet quartets. In: *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, New York, NY, USA: ACM, 2002, p. 293–305 (*IMW '02*, v.).
Disponível em http://doi.acm.org/10.1145/637201.637247 (Acessado em 5 jun. 2020)

[96] PAXSON, V.; ALLMAN, M. Rfc2988: Computing tcp's retransmission timer. 2000.

[97] PAXSON, V.; ALMES, G.; MAHDAVI, J.; MATHIS, M. *Framework for ip performance metrics*. RFC 2330, RFC Editor, 1998.

[98] PETERSON, L.; ANDERSON, T.; CULLER, D.; ROSCOE, T. A blueprint for introducing disruptive technology into the internet. *ACM SIGCOMM Computer Communication Review*, v. 33, n. 1, p. 59–64, 2003.

[99] PIRAUX, M.; DE CONINCK, Q.; BONAVENTURE, O. Observing the evolution of QUIC implementations. In: *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, 2018, p. 8–14.

[100] PORETSKY, S.; PERSER, J.; ERRAMILLI, S.; KHURANA, S. *Terminology for benchmarking network-layer traffic control mechanisms*. RFC 4689, RFC Editor, 2006.

[101] POSTEL, J. Transmission Control Protocol. RFC 793, 1981.

[102] PRASAD, R.; DOVROLIS, C.; MURRAY, M.; CLAFFY, K. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, v. 17, n. 6, p. 27–35, 2003.

[103] RABITSCH, A. *Evaluation of packet schedulers for multipath quic*. Dissertação de Mestrado, Karlstad University, Department of Mathematics and Computer Science (from 2013), 2018.

[104] RAICIU, C.; HANDLEY, M. J.; WISCHIK, D. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356, 2011.
Disponível em https://rfc-editor.org/rfc/rfc6356.txt (Acessado em 5 jun. 2020)

[105] RAMAKRISHNAN, K.; FLOYD, S.; BLACK, D. *The addition of explicit congestion notification (ecn) to ip*. Internet Standart RFC 3168, IETF, 2001.

[106] RESCORLA, E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.
Disponível em https://rfc-editor.org/rfc/rfc8446.txt (Acessado em 5 jun. 2020)

[107] RESCORLA, E.; DIERKS, T. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, 2008.
Disponível em https://rfc-editor.org/rfc/rfc5246.txt (Acessado em 5 jun. 2020)

[108] RESCORLA, E.; MODADUGU, N. Datagram Transport Layer Security Version 1.2. RFC 6347, 2012.
Disponível em https://rfc-editor.org/rfc/rfc6347.txt (Acessado em 5 jun. 2020)

[109] RIBEIRO, V. J.; RIEDI, R. H.; BARANIUK, R. G.; NAVRATIL, J.; COTTRELL, L. pathchirp: Efficient available bandwidth estimation for network paths. In: *Passive and active measurement workshop*, 2003.

[110] ROSKIND, J. Quic: Design document and specification rationale. 2012.
Disponível em https://goo.gl/eCYF1a (Acessado em 5 jun. 2020)

[111] RUBENSTEIN, D.; KUROSE, J.; TOWSLEY, D. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions on Networking*, v. 10, n. 3, p. 381–395, 2002.

[112] SALTZER, J. H. *On the naming and binding of network destinations*. RFC 1498, RFC Editor, http://www.rfc-editor.org/rfc/rfc1498.txt, 1993.
Disponível em http://www.rfc-editor.org/rfc/rfc1498.txt (Acessado em 5 jun. 2020)

[113] SANDRI, M.; SILVA, A.; ROCHA, L. A.; VERDI, F. L. On the benefits of using multipath tcp and openflow in shared bottlenecks. In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA)*, 2015, p. 9–16.
Disponível em doi.ieeecomputersociety.org/10.1109/AINA.2015.159 (Acessado em 5 jun. 2020)

[114] SAROIU, S.; GUMMADI, P. K.; GRIBBLE, S. D. Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments. In: *IEEE INFOCOM*, 2002, p. 1.

[115] SAVAGE, S.; CARDWELL, N.; ANDERSON, T. The case for informed transport protocols. In: *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, 1999, p. 58–63.

[116] SEEMANN, M.; CLEMENTE, L. A quic implementation in pure go. https://github.com/lucas-clemente/quic-go, 2020.

[117] SESHAN, S.; STEMM, M.; KATZ, R. H. *Spand: Shared passive network performance discovery*. Relatório Técnico, Berkeley, Berkeley, CA, USA, 1997.

[118] SODAGAR, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia*, v. 18, n. 4, p. 62–67, 2011.

[119] SOMMERS, J.; YEGNESWARAN, V.; BARFORD, P. A framework for malicious workload generation. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, New York, NY, USA: Association for Computing Machinery, 2004, p. 82–87 (*IMC '04*, v.).
Disponível em https://doi.org/10.1145/1028788.1028799 (Acessado em 5 jun. 2020)

[120] STALLINGS, W. *Snmp, snmpv2, snmpv3, and rmon 1 and 2*, v. 3. Addison-Wesley Reading, MA, 1999.

[121] STRAUSS, J.; KATABI, D.; KAASHOEK, F.; KAASHOEK, F. A measurement study of available bandwidth estimation tools. In: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, New York, NY, USA: ACM, 2003, p. 39–44 (*IMC '03*, v.).
Disponível em http://doi.acm.org/10.1145/948205.948211 (Acessado em 5 jun. 2020)

[122] SUNDARESAN, S.; FEAMSTER, N.; TEIXEIRA, R. Home network or access link? locating last-mile downstream throughput bottlenecks. In: KARAGIANNIS, T.; DIMITROPOULOS, X., eds. *Passive and Active Measurement*, Cham: Springer International Publishing, 2016, p. 111–123.

[123] TIRUMALA, A. Iperf: The tcp/udp bandwidth measurement tool. *http://dast. nlanr. net/Projects/Iperf/*, 1999.

[124] TTCP, U. A test of tcp and udp performance. 1984.

[125] VIERNICKEL, T.; FROEMMGEN, A.; RIZK, A.; KOLDEHOFE, B.; STEINMETZ, R. Multipath quic: A deployable multipath transport protocol. In: *2018 IEEE International Conference on Communications (ICC)*, 2018, p. 1–7.

[126] VU, V. A.; WALKER, B. On the latency of multipath-quic in real-time applications. In: *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2020, p. 1–7.

[127] WALID, A.; HWANG, J.; PENG, Q.; LOW, S. H. Balia (Balanced Linked Adaptation) – A New MPTCP Congestion Control Algorithm. In: *Proceedings of the 90th IETF Meeting*, Toronto, Ontario/Canada, 2014.
Disponível em https://www.ietf.org/proceedings/90/slides/slides-90-mptcp-0.pdf

[128] WEI, W.; WANG, Y.; XUE, K.; WEI, D. S. L.; HAN, J.; HONG, P. Shared bottleneck detection based on congestion interval variance measurement. *IEEE Communications Letters*, v. 22, n. 12, p. 2467–2470, 2018.

[129] Wei, W.; Xue, K.; Han, J.; Wei, D. S. L.; Hong, P. Shared bottleneck-based congestion control and packet scheduling for multipath tcp. *IEEE/ACM Transactions on Networking*, p. 1–14, 2020.

[130] White, K. *Definitions of managed objects for remote ping, traceroute, and lookup operations*. RFC 2925, RFC Editor, 2000.

[131] Wischik, D.; Raiciu, C.; Greenhalgh, A.; Handley, M. Design, implementation and evaluation of congestion control for multipath tcp. In: *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, USA: USENIX Association, 2011, p. 99–112 (*NSDI'11*, v.).

[132] Wydrowski, B.; Andrew, L. L. H.; Zukerman, M. Maxnet: a congestion control architecture for scalable networks. *IEEE Communications Letters*, v. 7, n. 10, p. 511–513, 2003.

[133] Zekauskas, M. J.; Karp, A.; Shalunov, S.; Boote, J. W.; Teitelbaum, B. R. A One-way Active Measurement Protocol (OWAMP). RFC 4656, 2006. Disponível em https://rfc-editor.org/rfc/rfc4656.txt (Acessado em 5 jun. 2020)

[134] Zhu, W. Tcp path sharing detection. In: *2013 IEEE 11th Malaysia International Conference on Communications (MICC)*, 2013, p. 244–249.

[135] Ziviani, A. An overview of internet measurements:fundamentals, techniques, and trends. *African J. of Inf. and Commun. Technology*, v. 2, 2006.